



Documentation for FishEye 4.0

Contents

Getting started	6
Supported platforms	7
Native support for SVN	11
Installing FishEye on Windows	16
Installing FishEye on Linux and Mac	19
Starting to use FishEye	22
Configuring JIRA integration in the Setup Wizard	27
Using FishEye	33
Using the FishEye screens	34
Browsing through a repository	35
Viewing file content	37
Viewing a file history	38
Using side by side diff view	39
Viewing the changelog	40
Searching FishEye	42
FishEye charts	46
Using favourites in FishEye	49
Changeset discussions	50
Viewing the commit graph for a repository	50
Viewing people's statistics	55
Using Smart Commits	57
Changing your user profile	62
Re-setting your password	65
Pattern matching guide	65
Date expressions reference guide	66
EyeQL reference guide	67
Integrating FishEye with Atlassian applications	73
JIRA Integration in FishEye	74
Integrating FishEye with Bitbucket Server	78
Transitioning issues in JIRA	78
Administering FishEye	80
Managing your repositories	81
Adding an external repository	82
CVS	83
Git	84
Mercurial	86
Perforce	87
Subversion	89
SVN fisheye.access	92
SVN tag and branch structure	93
Verifying SVN symbolic rules	101
Setting up a repository client	106
CVS client	106
Git client	106
Mercurial client	107
Perforce client	107
Subversion client	108
Native Subversion Client	108
SVNKit client	110
Re-indexing your repository	111
Repository options	112
Authentication	114
Checkout URLs	119
Commit message syntax	121
Configuring repository details	122
Hidden directories	125

Include Exclude paths	126
Indexer	130
JIRA workflow triggers	132
Linkers	132
Operations	136
Permissions	136
Properties	138
Store diff info	140
Tarball settings	140
Text file size limiting	142
Updater	143
Watches	144
Pipelined indexing	147
Managing users and groups in FishEye	148
Creating a user	149
Creating a group	149
Adding a user to a group	150
Associating a group with a repository	150
Configuring avatar settings	150
Configuring anonymous access	151
Editing a user's details	152
Changing a user's password	152
Renaming a user	152
Deleting or deactivating a user	153
Global permissions	153
External user directories	154
Connecting to an existing LDAP directory	156
Connecting to Crowd	163
Connecting to JIRA for user management	165
User management limitations and recommendations	167
Delegating authentication to an LDAP directory	169
Migrating to an external database	173
Migrating to MySQL	174
Migrating to Oracle	178
Migrating to PostgreSQL	181
Migrating to SQL Server	183
Configuring the database connection pool	186
Configuring SMTP	188
Customizing email notifications	189
Freemarker Data Model for Email Templates	190
Backing up and restoring FishEye data	192
Managing add-ons	199
Linking FishEye to JIRA	200
JIRA FishEye Plugin version compatibility	202
Linking to another application	202
System settings	203
FishEye folder layout	204
Blame calculations	205
Command-line options	207
Configuring web hooks	210
Enabling Smart Commits	214
Configuring commit hooks	216
Setting the REST API token	221
Configuring ViewVC compatibility	222
Customizing the front page	222
Configuring raw file download settings	224
Software update notifications	225
Enabling Access Logging in FishEye	225
FishEye and Crucible Cookies	226
Configuring the FishEye web server	229
FishEye SSL configuration	233
Integrating with other web servers	237

Running FishEye as a Windows service	239
Configuring indexing	243
Optimizing the Search Indexes on Restart	243
Customising FishEye's look and feel	244
Environment variables	245
Configuring outgoing web proxy support for FishEye or Crucible	249
Using AJP with FishEye	250
Custom authentication	251
Configuring SSL cipher suites for Jetty	252
JVM system properties	254
Configuring user managed mappings	261
Configuring FishEye security	261
Best practices for configuring FishEye security	261
Brute force login protection	263
Configuring CAPTCHA	264
Granting administrator privileges	264
Configuring public signup	264
Tuning FishEye performance	265
Best practices for FishEye configuration	268
Separating FishEye instance data from application data	269
Best practices Subversion integration	270
Troubleshooting SVN indexing issues in FishEye and Crucible	271
How FishEye uses memory	273
Contacting Support	275
Finding your FishEye or Crucible Support Entitlement Number (SEN)	279
FishEye FAQ	279
General FAQ	281
About database encoding	282
About the Lines of Code Metric	283
Cannot View Lines of Code Information in FishEye	285
Finding your Server ID	285
How Do I Archive a Branch within Perforce	285
How do I Avoid Long Reindex Times When I Upgrade	286
Mercurial Known Issues	288
Ordering of Branches Important When Visualizing Git Changesets	288
Permanent authentication for Git repositories over HTTP(S)	289
Perforce Changesets and Branches	290
What SCM systems are supported by FishEye?	290
Automating Administrative Actions in Fisheye	290
How are indexing requests handled when they are triggered via commit hook	291
Installation & Configuration FAQ	291
Can I deploy FishEye or Crucible as a WAR?	292
Does Fisheye support SSL (HTTPS)?	292
Improve FishEye scan performance	292
Migrating FishEye Between Servers	293
Setting up a CVS mirror with rsync	294
What are the FishEye System Requirements?	295
How to reset the Administration Page password in FishEye or Crucible	296
How Do I Configure an Outbound Proxy Server for FishEye	296
How to remove Crucible from FishEye 2.x or later	296
How to run Fisheye or Crucible on startup on Mac OS X	297
Licensing FAQ	298
Are anonymous users counted towards FishEye's license limits?	298
Restrictions on FishEye Starter Licenses	299
Updating your FishEye license	301
Git or Hg Repository exceeds number of allowed committers	302
Example EyeQL Queries	304
How do find changes made to a branch after a given tag?	304
How do I filter results?	304
How do I find changes between two versions, showing separate histories?	305
How do I find changes made between two version numbers?	305
How do I find commits without comments?	305

How do I find files on a branch, excluding deleted files?	305
How do I find files removed from a given branch?	305
How do I find revisions made by one author between versions?	305
How do I select the most recent revisions in a given branch?	306
How do I show all changesets which do not have reviews?	306
Integration FAQ	306
How do I disable the Source (FishEye) tab panel for non-code projects?	306
How do I enable debug logging for the JIRA FishEye plugin?	306
How do I uninstall the JIRA FishEye plugin?	307
How is the Reviews (Crucible) tab panel for the JIRA FishEye Plugin populated?	307
What do I do if I discover a bug with the JIRA FishEye plugin?	307
Subversion FAQ	307
Configuring Start Revision based on date	308
Errors 'SEVERE assert' or 'Checksum mismatch'	308
FishEye fails to connect to the Subversion repository after a short time of successful operation.	308
How can FishEye help with merging of branches in Subversion?	308
Subversion Changeset Parents and Branches	309
SVN Authentication Issues	309
What are Subversion root and tag branches?	310
Why do I need to describe the branch and tag structure for Subversion repositories? ...	310
Why don't all my tags show up in FishEye?	311
About merges in Subversion	313
CVS FAQ	316
How does FishEye calculate CVS changesets?	316
How is changeset ancestry implemented for CVS?	316
Support Policies	316
Bug Fixing Policy	316
New Features Policy	317
Security Bugfix Policy	318
Troubleshooting	318
After I commit a change to my CVS repository, it takes a long time before it appears in FishEye.	319
FishEye freezes unexpectedly	319
I have installed FishEye, and the initial scan is taking a long time. Is this normal?	320
I have installed FishEye, but there is no data in the Changelog.	320
Initial scan and page loads are slow on Subversion	321
JIRA Integration Issues	321
Manually Generating a Thread Dump	321
Message 'org.tigris.subversion.javahl.ClientException svn Java heap space'	324
Problems with very long comments and MySQL migration	324
URLs with encoded slashes don't work, especially in Author constraints	324
FishEye Developer FAQ	325
Contributing to the FishEye Documentation	325
FishEye Documentation in Other Languages	326
FishEye Resources	327
Glossary	327
Collecting analytics for FishEye	329

Getting started

Atlassian's FishEye is the on-premise source code repository browser for enterprise teams. It provides your developers with advanced browsing and search for SVN, Git, Mercurial, Perforce and CVS code repositories, from any web browser.

Browse repos from your browser!

- SVN, Hg, Git, CVS, P4
- files, changesets, revisions, branches, tags, diffs, annotations
- side-by-side diffs
- unified diffs for word-level changes
- see details at the repository, branch, directory, or file levels
- filter commits by log message, path, author, date, branch

Search everything

- File names, commit messages, authors, text, JIRA issue keys, partial paths, wild cards, camel case
- QuickNav
- Simple search
- Advanced search - SQL-like syntax (EyeQL)
- Bookmarks any query
- Download search results
- Link to any revision, changeset, diff view, line of code, search

Monitor and visualize

- Understand what, when, where, who, how changes were introduced
- Commit graph for changesets
- Activity streams
- Charts and reports
- Notifications for code activity byemail, RSS, OpenSocial dashboards

Integrate with JIRA and other Atlassian applications

- See related JIRA issues
- Use smart commits to transition issues
- Crucible reviews
- Bamboo builds

Enterprise ready

- User management and authentication
- Database support and migration
- Security
- Performance
- Backup and restore

Ecosystem

- Marketplace for add-ons
- REST API to write your own custom add-ons e.g. to integrate with other applications

Get started!

1. Install and start FishEye on either [Windows](#), or [Linux and Mac](#).
2. Work through [Starting to use FishEye](#).
3. Tell FishEye about your [repositories](#).
4. Set up [users and groups](#).

Supported platforms

This page lists the supported platforms for **FishEye 4.0.x** and its minor releases.

Key:  = Supported  = Deprecated  = Not Supported

Java		
Oracle JRE / JDK	 1.8  1.7	FishEye requires the Java Runtime (JDK or JRE), version as noted. Pre-release/Early access versions of Java are <i>not supported</i>.
OpenJDK	 1.8 (Linux only)	We highly recommend that you use the Oracle JVM (or OpenJDK for Linux only). Other Java implementations have not been tested. You can download an Oracle Java Runtime. For the OpenJDK, download and install instructions for Linux flavors are at http://openjdk.java.net/install/. Please note: • Once you have installed Java, you must set the JAVA_HOME environment variable. See Installing FishEye on Windows or Installing FishEye on Linux and Mac. • If you are using a 64-bit JVM, please ensure that you've set your max heap size (-Xmx) to a reasonable value, considering the RAM requirements of your system. • If you intend to run FishEye as a Windows Service, using the Java Service Wrapper, you should use the Java JDK rather than the JRE so as to take advantage of the -server parameter. If you're using the Windows Installer distribution, which handles the FishEye service differently, using the Java JRE should suffice. • You'll need the JDK for the JSP source download. • For OpenJDK, you'll need the DejaVu font package installed. Installation instructions can be found here: http://dejavu-fonts.org/wiki/Download.  Support for Java 7 was removed in FishEye 3.9, as previously announced.
Operating systems		
Microsoft Windows		• FishEye is a pure Java application and should run on any platform provided the requirements for the JRE or JDK are
Linux		

Apple Mac OS X		satisfied. Although FishEye can be run in virtualized environments, Atlassian is not yet able to provide technical support for performance-related problems in a virtualized environment. If you do choose to run FishEye in a VM, please ensure that you choose a VM with good IO throughput.
Databases		
HSQldb	Bundled; for evaluation use only	The FishEye built-in database, running HSQldb, is somewhat susceptible to data loss during system crashes. We recommend that you do not use HSQldb for production systems. External databases (such as MySQL) are generally more resistant to data loss during a system crash. See the FishEye Database documentation for further details.
MySQL	MySQL Enterprise Server 5.1+ MySQL Community Server 5.1+ MySQL 5.0 MariaDB, Percona	For MySQL: - For 5.1, versions earlier than 5.1.10 are not supported - For 5.6, versions earlier than 5.6.11 are not supported - For 5.7, versions earlier than 5.7.5 are not supported - Support for MySQL 5.0 was removed in FishEye 3.3. See End of Support Announcements for FishEye. - MariaDB and Percona variants of MySQL are not supported, and are known to cause issues when used with FishEye. Support for PostgreSQL 8.2 was removed in FishEye 3.3. See End of Support Announcements for FishEye. Support for SQL Server 2005 was removed in FishEye 3.3. See End of Support Announcements for FishEye.
PostgreSQL	9.0, 9.1, 9.2, 9.3, 9.4 8.3, 8.4 8.2	
Oracle	12c 11g	
SQL Server	2012 2008, 2008 R2 2005	
Web browsers		
Microsoft Internet Explorer	10.0, 11.0 9.0	Support for Internet Explorer 9 was removed in FishEye 3.9. See End of Support Announcements for FishEye.
Mozilla Firefox	Latest stable version supported 3.6, 4.0	Support for Firefox 3.6 and 4.0 was removed in FishEye 3.7.
Safari	Latest stable version supported 4, 5	Support for Safari 4 and 5 was removed in FishEye 3.7.

Chrome	 Latest stable version supported	
Version control systems		
Subversion	Server:  1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8 Client:  SVNKit (bundled & the default)  Native JavaHL 1.8  Native JavaHL 1.7  Native JavaHL 1.6	 FishEye 3.1, and later, do not support the native JavaHL 1.6 client. See Native support for SVN for discussion.
CVS (and CVSNT)	 All versions	
Perforce	 Client version 2007.3 or later	The Server must support the ztag tagged protocol.  Perforce Streams, introduced in 2011.1, is not currently supported. See FE-3886 - support for Streams in p4 OPEN
Git	 2.8.2  2.7.3  2.6.5  2.5.4  2.4.10  2.3.10  2.2.3  2.1.4  2.0.5  1.9.5  1.8.0.3, 1.8.1.5, 1.8.2.3, 1.8.3.4, 1.8.4.5, 1.8.5.6  1.7.1.1, 1.7.2.5, 1.7.3.5, 1.7.4.5, 1.7.5.4, 1.7.6.6, 1.7.7.7, 1.7.8.6, 1.7.9.7, 1.7.10.5, 1.7.11.7, 1.7.12.4	These are the versions of Git that we currently test FishEye against.  Git 1.8.4.3 is not supported. See BSERV-4101 - Clone and fetch fail with "protocol error: impossibly long line" CLOSED Externally-hosted repositories only are supported. [Security vulnerability CVE-2014-9390] affects multiple Git versions. FishEye itself is not affected, however you should update your <i>clients</i> to a patched maintenance version: v1.8.5.6, v1.9.5, v2.0.5, v2.1.4 and v2.2.1 or newer. Git for Windows is the <i>only supported distribution</i> when running FishEye on Windows. Cygwin Git is <i>not supported</i> and has known issues.

Mercurial	✔ 3.0.2, 3.1.2, 3.2.4, 3.3.3, 3.4.2, 3.5.2, 3.6.3, 3.7.2 ✔ 2.0.2, 2.1.2, 2.2.3, 2.3.2, 2.4, 2.5.2, 2.6.3, 2.7.2, 2.8.2, 2.9.1 ✔ 1.5.1, 1.5.4, 1.6.4, 1.7.5, 1.8.4, 1.9.3	These are the versions of Mercurial that we currently test FishEye against. As of version 3.6.3, FishEye supports Mercurial 3. ✖ Mercurial 2.1 has a bug that makes it incompatible with FishEye. Please use Mercurial 2.1.1 or later. You should restart FishEye after upgrading Mercurial.
Atlassian applications		
Crowd	✔ Crowd 2.4.x+ ✔ Crowd client library: 2.4.1	From version 2.8.x, FishEye bundles the Crowd 2.4.1 client library, and supports the Crowd 2.4.x server, and later versions.
JIRA	✔ JIRA 5.0+	FishEye to JIRA communication requires JIRA 5.0.x or later. Communication the other way, from JIRA to FishEye, depends on the JIRA FishEye Plugin. Note that the JIRA FishEye Plugin is bundled with JIRA. If you are using a version of JIRA earlier than JIRA 5.0 you can upgrade the plugin in JIRA to get support for FishEye.

Hardware requirements

FishEye should ideally run on a dedicated server. The most important aspect for a large-repository deployment will be I/O speed. You definitely want a fast local HDD for FishEye's cache. Note that NFS and SAN are not supported.

Component	Specifications
CPU	1.8GHz or higher, a single core is sufficient. More cores or higher GHz will result in better load-handling ability.
RAM	1GB minimum, 2GB will provide performance "headroom". Your Java heap should be sized at 512MB with the <code>FISHEYE_OPTS</code> environment variable, adjustable up to 1024MB depending on performance.
I/O	FishEye's input/output is an important element of its overall performance. If FishEye accesses your repository remotely, make sure that the throughput is maximum and the latency minimum (ideally the servers are located in the same LAN, running at 100Mbps or faster).
Monitor	Minimum screen resolution of 1024x768. Recommended screen resolution of 1280x768 or above.

While some of our customers run FishEye on SPARC-based hardware, Atlassian only officially supports FishEye running on x86 hardware and 64-bit derivatives of x86 hardware.

Disk space requirement estimates

Disk space requirements for FishEye may vary due to a number of variables such as the repository implementation, file sizes, content types, the size of diffs and comments being stored. The following table contains some real-world examples of FishEye disk space consumption.

Repository technology	Commits	Codebase size (HEAD of trunk)	FishEye index size
Subversion	14386	466 MB in 12151 files	647 MB
CVS	8210	115 MB in 11433 files	220 MB

 These disk space estimates are to be used as a guideline only. We recommend you monitor the disk space that your FishEye instance uses over time, as needs for your specific environment may vary. It may be necessary to allocate more space than indicated here. Additionally, you can reduce disk space consumption by [turning off diff storage](#) in FishEye.

Deployment notes for version control systems

Subversion (server)	FishEye can communicate with any repository running Subversion 1.1 or later.
Subversion (client)	FishEye now bundles the SVNKit client, which becomes the default Subversion interface. An alternative is to use the native subversion client, using JavaHL bindings. Please see Subversion Client Setup for more information.
Perforce (client)	FishEye needs access to the p4 client executable. Due to some problems with earlier versions of the client, we recommend version 2007.3 or later.
CVS	If you are using CVS, FishEye needs read-access to your CVS repository via the file system . It does not support protocols such as pserver at the moment.

Support for other version control systems is planned for future releases. Let us know what SCM system you would like to see supported by creating a [JIRA issue](#) or adding your vote to an issue, if the request already exists.

WAR deployment

FishEye/Crucible is a standalone Java program. It cannot be deployed to web application servers such as WebSphere, Weblogic or Tomcat.

Single sign on with Atlassian Crowd

From version 2.8.x, FishEye bundles the Crowd 2.4.1 client library, and supports the Crowd 2.4.x server, and later versions.

Native support for SVN

This page describes an advanced feature of FishEye's Subversion support. It explores the technical background, and some of the issues you may encounter, if you wish to use the native JavaHL access feature.

For most users we recommend that you use the default SVNKit Subversion access client that is bundled with FishEye. You are only likely to need the native JavaHL access described on this page for certain edge case repositories.

FishEye Subversion access

FishEye interacts with Subversion repositories through a layer, defined by the Apache Subversion project, known as JavaHL. This is the high-level Java language binding for Subversion. There are two implementations of the JavaHL interface available:

Bundled SVNKit

The SVNKit implementation is a largely Java-based implementation provided by the SVNKit project. It is bundled with FishEye and is the default JavaHL implementation used. As a Java implementation it operates on all of FishEye's supported platforms.

Native JavaHL

The JNI-based implementation, coupled with a shared dynamic library, is referred to here as native JavaHL. As a native library, native JavaHL is platform-dependent. The shared library is C-based and must be compatible with the remaining Subversion client components installed on the platform. It varies across each platform and distribution.

On this page:

- [FishEye Subversion access](#)
- [Native vs. bundled](#)
- [Native JavaHL support](#)
- [Installing JavaHL for your platform](#)
 - [Windows 7](#)
 - [Ubuntu 12](#)
 - [CentOS 6.4](#)
- [JavaHL considerations when upgrading FishEye](#)

Native vs. bundled

Given that FishEye bundles the SVNKit implementation, why might you want to use the native implementation of JavaHL? In general our recommendation is to stick with the bundled SVNKit implementation. It is the simplest to use and works in the widest variety of scenarios. Nevertheless, there are some scenarios where it may be desirable to use the native implementation, if it is available.

The two implementations have quite different characteristics – these can affect the decision about which to use. Here is a high-level list of some of the considerations we have encountered over the years:

Aspect	SVNKit - bundled with FishEye	Native JavaHL - platform dependent
Memory Usage	SVNKit uses the Java Heap. It therefore shares the heap that is being used for all FishEye's operations. It does benefit, however, from Java's garbage collection mechanism and we have not seen any memory leaks.	The native JavaHL implementation uses the native process heap and not the Java heap. It can increase the overall process memory usage but does not interfere with the Java heap usage. In some rare instances, we have seen memory leaks in the C-based JavaHL code. As FishEye is a long running service, these can cause problems over the life of the FishEye process.
Speed	In general, when using any of the Subversion network protocols, the JavaHL implementation speed is not a significant factor in the overall speed of Subversion operations as they are dominated by the network latency. Even for file:// access SVNKit is rarely the bottleneck.	If you are using file:// access to talk to a Subversion repository on the same server, then native JavaHL will most likely give the highest performance.
Compatibility	SVNKit has proven to be highly compatible with Subversion across all releases. The project is very responsive to bug reports when any differences become apparent. As an alternative implementation of JavaHL there will be differences between the SVNKit and the native Subversion JavaHL. This may affect some edge case repositories.	JavaHL uses predominantly the same code as Subversion itself so it is virtually 100% compatible.

Availability	SVNKit works on all of FishEye's supported platforms	It can be difficult to get an install of the JavaHL jar and shared library that is compatible with the version of Subversion installed on your platform.

Native JavaHL support

The native JavaHL interface and implementation naturally change with every release of Subversion. Normally these changes are incremental and backward compatible.

The compatibility matrix for recent FishEye versions is:

	SVNKit	Native JavaHL 1.6 client	Native JavaHL 1.7 client	Native JavaHL 1.8 client
FishEye 3.3+				
1.8 Subversion Server	✓	✗	✓	✓
1.7 Subversion Server	✓	✗	✓	✓
1.6 Subversion Server	✓	✗	✓	✓
FishEye 3.1 – 3.2				
1.8 Subversion Server (not file:// access)	✓	✗	✓	Unsupported
1.8 Subversion Server (file:// access)	✗	✗	✗	Unsupported
1.7 Subversion Server	✓	✗	✓	Unsupported
1.6 Subversion Server	✓	✗	✓	Unsupported
FishEye 3.0 and earlier				
1.7 Subversion Server (not file:// access)	✓	✓	✗	✗
1.7 Subversion Server (file:// access)	✓	✗	✗	✗
1.6 Subversion Server any access	✓	✓	✗	✗

Note that:

- FishEye 3.1, and later, is not compatible with the JavaHL 1.6 client (or older versions) – it uses the org.apache.subversion package which is not provided in 1.6 JavaHL builds
- From FishEye 3.1, you must use a 1.7 or later JavaHL library if you want native JavaHL access.

✓ [Click here to read about compatibility changes from SVN 1.6 to 1.7...](#)

The change from Subversion 1.6 to Subversion 1.7 was much more significant and for a number of FishEye's usages of the interface, it broke compatibility.

In Subversion 1.7, the JavaHL interfaces were updated and moved from the org.tigris.subversion package to the org.apache.subversion package. This coincided with the move of the Subversion project to the Apache Software Foundation. In addition to moving the package, the interface was modernized in a number of ways:

- Extended use of callbacks.
- Use of Java collections rather than native arrays.
- Properties were clarified as byte arrays rather than Java Strings.
- Use of typed Enums rather than primitive integer and char fields.

The existing org.tigris package was retained in most 1.7 distributions and was implemented as an adapter

layer over the new org.apache package classes. Unfortunately a number of incompatibilities in the adapter layer meant that FishEye could not use the 1.7 native implementation:

1. Property returning methods were wrapped in a String constructor to convert from the byte[] type of the new interface to the String type used in the old interface. This meant that any null returns would throw NullPointerExceptions rather than returning null Strings.
2. Some of the callbacks were changed from plain interfaces to being interface extensions of the corresponding callback in the new package. This changed the type definition of the callback from an untyped Map to a typed Map. This caused ClassCastExceptions because the code is expecting a map containing byte[] but the underlying code was passing in a map containing strings.

For this reason, FishEye did not support native access using the 1.7 native library prior to FishEye 3.1.

Installing JavaHL for your platform

Atlassian FishEye bundles the SVNKit library to make connecting to your Subversion repository a painless process out of the box. If you do wish to use native JavaHL, it is your responsibility to install it onto your platform. Different organizations have different operating procedures and policies regarding how and what packages they are able to install on production servers.

In some cases the distribution you use will not provide a compatible JavaHL from an official package. In this case you will either need to build everything from source yourself (hard) or use a package from a Subversion vendor. We have used packages from two vendors over time, CollabNet and Wandisco. More recently, we have found it easier to use the Wandisco packages for JavaHL support.

The following sections detail our experiences when we investigated deploying JavaHL 1.7 on a variety of platforms. This is not a definitive list or guide. It is to give you an idea of some of the issues you are likely to encounter getting a compatible JavaHL install working on a range of platforms and distributions.

32-bit / 64-bit: The DLL file is platform dependent so it needs to match the architecture of your VM. The JAR file however, contains only metadata for the JVM about how to load the DLL so the same JAR will work across 32-bit and 64-bit operating systems as long as the subversion binaries match.

Windows 7

Windows does not include a Subversion client by default so you will need to install a Subversion package. We installed the 1.7.11 "client only" install from Wandisco. This installs Subversion, including the javahl components, in C:\Program Files\WANDisco\Subversion. It is interesting to note that the JavaHL package in this install does not include the org.tigris package adapter layer.

Ubuntu 12

Ubuntu provides packages for both core subversion and the JavaHL library for Subversion. We installed these for Subversion 1.7.5 as follows:

- `sudo apt-get install subversion`
- `sudo apt-get install libsvn-java`

Unfortunately the version installed seems to have a consistent assertion failure:

```
java: /build/buildd/subversion-1.7.5/subversion/libsvn_subr/dirent_uri.c:1519:
uri_skip_ancestor: Assertion `svn_uri_is_canonical(child_uri, ((void *)0))' failed.
```

We then removed the two Subversion packages from Ubuntu itself:

- `sudo apt-get remove libsvn-java`
- `sudo apt-get remove subversion`

We installed the Wandisco packages by downloading and running the Wandisco installer:

```
svn1.7_ubuntu_wandisco-precise.sh
```

This configures the Wandisco servers as a source of packages and installs the core Subversion install. At the time of writing this installed 1.7.11. Once installed, reinstall the javahl package:

- `sudo apt-get install libsvn-java`

This will now come from the Wandisco package repository. The location of the shared library and JavaHL jar is:

```
/usr/lib/jni/libsvnjavahl-1.so
/usr/share/java/svn-javahl.jar
```

CentOS 6.4

If you install the Subversion packages (subversion and subversion-javahl) using yum, you will have a 1.6.11 install of Subversion which is not compatible with FishEye if you wish to use JavaHL as described above.

If you have previously upgraded to a version of SVN 1.7 before 1.7.11 you may see the message below in your logs. If you do, please upgrade to the Wandisco SVN 1.7 as described below:

```
java: /build/buildd/subversion-1.7.5/subversion/libsvn_subr/dirent_uri.c:1519:
uri_skip_ancestor: Assertion `svn_uri_is_canonical(child_uri, ((void *)0))' failed.
```

You will need to remove the standard yum packages and use a Wandisco install, `svn1.7_centos6_wandisco.sh`. This installs plain subversion and configures the Wandisco servers as a source of packages. You can then use yum to install subversion-javahl. The following files are installed:

```
$ repoquery --list subversion-javahl
/usr/lib/libsvnjavahl-1.so
/usr/lib/svn-javahl/svn-javahl.jar
/usr/lib64/libsvnjavahl-1.so
/usr/lib64/svn-javahl/svn-javahl.jar
```

If you are using a 64bit JVM, use the `/usr/lib64` library, otherwise use the 32bit library in `/usr/lib`.

JavaHL considerations when upgrading FishEye

If you are currently using SVNKit with FishEye (the default), then you do not have to do anything when upgrading to FishEye 3.1 and later. FishEye will continue to use the bundled SVNKit library.

FishEye's Admin UI now displays information about the Subversion client in use – click **Server**, under 'Global Settings'. With no native client, configured, the display would look like:

Subversion client

The bundled Subversion client, SVNKit, is being used for Subversion operations.
The JavaHL client version is SVNKit v1.8.3.10190.

JAR *not set*

Dynamic library *not set*

Edit Details

If you have been using native JavaHL prior to FishEye 3.1, FishEye will detect that you have configured a pre-1.7 version of JavaHL and fallback to the bundled SVNKit client and start up normally. You will see the following in the **Server** section of the admin UI:

Subversion client

The configured Subversion client, a pre 1.7 native JavaHL client, is no longer supported.
The bundled Subversion client, SVNKit, is being used for Subversion operations.
Please update the configuration to a 1.7 or later version to continue using the native JavaHL Subversion client.

JAR `/opt/subversion/lib/svn-javahl/svn-javahl.jar`

Dynamic library `/opt/subversion/lib/libsvnjavahl-1.0.dylib`

[Edit Details](#)

You can use the FishEye admin UI to update the JavaHL client information to point FishEye to a 1.7 or later JavaHL jar and shared library. FishEye will perform some checks that the configured library supplies the correct classes. You will need to restart for the changes to take effect. If there are problems with the JavaHL library on restart, FishEye will again fallback to SVNKit. Once you have updated the configuration, FishEye will show a message that the configuration has been changed and a restart is required:

Subversion client

The Subversion client configuration has been changed and will take effect after restart.

JAR `not set`

Dynamic library `not set`

[Edit Details](#)

Upon restart, the display will show the operation of the native library and its version:

Subversion client

The configured Subversion client, native JavaHL, is being used for Subversion operations.
The JavaHL client version is 1.8.0 (r1490375).

JAR `/opt/subversion/lib/svn-javahl/svn-javahl.jar`

Dynamic library `/opt/subversion/lib/libsvnjavahl-1.0.dylib`

[Edit Details](#)

Installing FishEye on Windows

This page...

... describes how to perform a clean install of FishEye on Windows.

Upgrading?

If you're upgrading your FishEye installation, read the [FishEye upgrade guide](#) first.

Using Linux or Mac OS X?

If you're on one of these platforms read [Installing FishEye on Linux and Mac ins](#)

tead.

1. Check supported platforms

Better check the [Supported platforms](#) page first; it lists the application servers, databases, operating systems, web browsers and JDKs that we have tested FishEye with, and that we recommend.

Atlassian only officially supports FishEye running on x86 hardware and 64-bit derivatives of x86 hardware.

2. Create a dedicated FishEye user (recommended)

For production installations, we recommend that you create a new dedicated Windows user that will run FishEye on your system. This user:

- Should not have admin privileges.
- Should be a non-privileged user with read, write and execute access on the FishEye install directory and instance (data) directory. These directories are described below.
- Should only have read access to your repositories.

If you have created a dedicated FishEye user, ensure you are logged in as this user to complete the remaining instructions.

3. Check your version of Java

In a command prompt, run this:

```
java -version
```

The version of Java should be **1.8.x**.

The recommended way to install FishEye is to use the installer, which installs FishEye as a Windows service – see step 5 below.

▼ [If you don't see a supported version of Java, then get Java...](#)

Download and install the Java Platform JDK from [Oracle's website](#).

 *We recommend that the Java install path should not contain spaces, so don't install into C:\Program Files\Java\. Instead, use a path like C:\Java.*

Now try running 'java -version' again to check the installation. The version of Java should be **1.8.x**.

4. Check that Windows can find Java

Windows uses the JAVA_HOME environment variable to find Java. To check that, in a new command prompt, run:

```
echo %JAVA_HOME%
```

You should see a path to the Java install location. We recommend that this path does *not* contain spaces, and that JAVA_HOME should point to the JDK home path.

▼ [If you don't see a path without spaces...](#)

- If you see a path with spaces, like `C:\Program Files\Java\`, then sorry, but go back to 3. and reinstall Java to a location that doesn't have spaces.
- If you don't see a path at all, or if you just see `%JAVA_HOME%`, then set `JAVA_HOME` as follows:

For Windows 7:

1. Go to **Start**, search for "sys env" and choose **Edit the system environment variables**.
2. Click **Environment Variables**, and then **New** under 'System variables'.
3. Enter "JAVA_HOME" as the **Variable name**, and the absolute path to where you installed the Java JDK as the **Variable value**, that is, something like `C:\Java\jdk1.7.0_51`. Don't use a trailing backslash. We recommend that `JAVA_HOME` should point to the JDK home path.
4. Now, in a new command prompt, try running '`java -version`'. You should see the same version of Java as you saw above.

5. Now it's time to get FishEye

Download the FishEye installer from the Atlassian download site.

There are 32-bit and 64-bit installers for FishEye on Windows. Each installer adds FishEye as a Windows service, and starts the service, automatically. The express install creates, by default, a Data directory and a separate install directory in `C:\Atlassian`. The custom install mode allows you to choose different locations for the install and Data directories, with the restriction that the Data directory must not be contained in the install directory.

- The installer creates the `FISHEYE_INST` system environment variable. This points to the location of the instance (data) directory.
- The path to the installation location is referred to as the `<FishEye install directory>` in these instructions.
- You need separate FishEye instance (data) directories if you want to run multiple copies of FishEye.
- If you expect to have a large number of users for this FishEye installation, and FishEye will be [connected to an external database](#), consider installing FishEye on a different server from the one running the external database, for improved performance.
- If you have a large number of repositories, we recommend you increase the default number of files that FishEye is allowed to open. See the following knowledge base article for more info: [Subversion Indexer Paused with "Too many open files" Error](#).
- For FishEye 3.4.4 and later, you can edit JVM parameters for the Windows service by going to **Start > All Programs > FishEye > Configure FishEye**. Ensure that you restart the FishEye service when finished. Do *not* reference any environment variables in the settings (e.g. `%FISHEYE_INST%`). Instead, set the actual path.

6. Visit FishEye!

Give the FishEye service a minute to launch. Then, in a web browser on the same machine, go to `http://localhost:8060/` (or, from another machine, type `http://hostname:8060/`, where `hostname` is the name of the machine where you installed FishEye).

Enter your license, then an admin password, to finish the setup. Note that this password is for the 'built-in' FishEye admin user. You can log in as this user, if necessary, by clicking the **Administration** link in the page footer. See also [How to reset the Administration Page password in FishEye or Crucible](#).

You can postpone setting up JIRA integration until later if you wish; see [Configuring JIRA integration in the Setup Wizard](#).

7. Add repositories

Now you can tell FishEye about any existing repositories you have. Please read [Starting to use FishEye](#) for the details.

FishEye will perform an initial index of your repositories, during which it accesses, indexes and organizes a view of your repositories (including all historical items) back to the earliest commits. If you are evaluating

FishEye, we suggest that you index a single project, so you can use FishEye as soon as possible. If you choose to index your entire repository, be aware that this can take a long time (possibly days) for massive or complex repositories and can be more complex to set up (especially for Subversion). The basic process is slightly different for each SCM type.

8. Add users and groups

You will want to set up your users and groups in FishEye. You can [add users directly](#) to FishEye, or connect to an [external user directory](#). Please read [Starting to use FishEye](#) for an introduction.

9. Set up your mail server

Configure the FishEye email server so that users can get notifications from FishEye. See [Configuring SMTP](#).

10. Connect to an external database (recommended)

If you intend to use this FishEye installation in a production environment, it is highly recommended that you use one of the [supported](#) external databases. See [Migrating to an external database](#).

If you are evaluating FishEye, or don't wish to do this now, FishEye will happily use its embedded HSQL database, and you can easily migrate later.

11. Stop FishEye (optional)

Control the FishEye service from the Windows administration console. Alternatively, in a command prompt, change directory to <FishEye install directory> and run this:

```
bin\stop.bat
```

12. Tuning FishEye performance

To get the best performance from your new FishEye installation, please consult [Tuning FishEye performance](#).

Installing FishEye on Linux and Mac

Hey! We're going to install FishEye on a Linux box, or a Mac. There are a few steps involved, but we think you'll find it easy to follow along. If you are upgrading an existing installation, please refer to the [FishEye upgrade guide](#) instead.

1. Check supported platforms

Better check the [Supported platforms](#) page first; it lists the application servers, databases, operating systems, web browsers and JDKs that we have tested FishEye with, and that we recommend.

Atlassian only officially supports FishEye running on x86 hardware and 64-bit derivatives of x86 hardware.

Related pages:

- [Installing FishEye on Windows](#)
- [Starting to use FishEye](#)
- [Supported platforms](#)
- [FishEye upgrade guide](#)

2. Create a dedicated FishEye user (recommended)

For production installations, we recommend that you create a new dedicated user that will run FishEye on your system. This user:

- Should not have admin privileges.
- Should be a non-privileged user with read, write and execute access on the FishEye install directory and instance (data) directory. These directories are described below.
- Should only have read access to your repositories.

If you created a dedicated FishEye user, ensure you are logged in as this user to complete the remaining instructions.

3. Check your version of Java

In a terminal, run this:

```
java -version
```

The version of Java should be **1.8.x**.

▼ If you don't see a supported version of Java, then get Java...

Download and install the [Oracle JVM](#) (JDK or JRE), or [OpenJDK](#).

Now try running 'java -version' again to check the installation. The version of Java should be **1.8.x**.

4. Check that the system can find Java

In a terminal, run this:

```
echo $JAVA_HOME
```

You should see a path like /System/Library/Frameworks/JavaVM.framework/Versions/CurrentJDK/Home/.

▼ If you don't see a path to the Java location, then set JAVA_HOME...

Linux

Do either of the following:

- If JAVA_HOME is not set, log in with 'root' level permissions and run:

```
echo JAVA_HOME="path/to/JAVA_HOME" >> /etc/environment
```

where path/to/JAVA_HOME may be like: /System/Library/Frameworks/JavaVM.framework/Versions/CurrentJDK/Home/

- If JAVA_HOME needs to be changed, open the /etc/environment file in a text editor and modify the value for JAVA_HOME to:

```
JAVA_HOME="path/to/JAVA_HOME"
```

It should look like:

```
JAVA_HOME=/System/Library/Frameworks/JavaVM.framework/Versions/CurrentJDK/Home/
```

5. Now it's time to get FishEye

1. [Download FishEye](#) from the Atlassian download site.

2. Please check your unzip program before extracting the downloaded zip file. Some archive-extract programs cause errors when unzipping the FishEye zip file:
 - Windows users must avoid the Windows built-in unzip utility, as it doesn't extract all the files. Use a third-party unzip program like [7-Zip](#) or [Winzip](#).
 - Solaris users will need to use [GNU tar](#) to handle the long file names.
3. Extract the downloaded file to an install location:
 - Folder names in the path to your FishEye executable should not have spaces in them. The path to the extracted directory is referred to as the `<FishEye install directory>` in these instructions.
 - If you expect to have a large number of users for this FishEye installation, and FishEye will be [connected to an external database](#), consider installing FishEye on a different server from the one running the external database, for improved performance.

6. Tell FishEye where to store your data

The FishEye instance directory is where your FishEye data is stored.

 You *should not* locate your FishEye instance directory inside the `<FishEye install directory>` — they should be entirely separate locations. If you do put the instance directory in the `<FishEye install directory>` it will be overwritten, and lost, when FishEye gets upgraded. And by the way, you'll need separate FishEye instance directories if you want to run multiple copies of FishEye.

For production installations, we recommend that the FishEye instance directory be [secured against unauthorized access](#).

Create your FishEye instance directory, and then tell FishEye where you created it by adding a `FISHEYE_INST` environment variable as follows:

Linux	Mac
Open the <code>/etc/environment</code> file in a text editor and insert: <code>FISHEYE_INST="path/to/<FishEye instance directory>"</code>	Open the <code>~/.profile</code> file for the current user in a text editor and insert: <code>FISHEYE_INST="path/to/<FishEye instance directory>"</code> <code>export FISHEYE_INST</code>
You need to log out and log in again so the new environment variable is set.	
Alternatively you can run <code>export FISHEYE_INST="path/to/<FishEye instance directory>"</code> to avoid logging out and in.	

Now, copy the `<FishEye install directory> /config.xml` to the root of the `FISHEYE_INST` directory, so that FishEye can start properly.

Also, if you have a large number of repositories, we recommend you increase the default number of files that FishEye is allowed to open. See the following knowledge base article for more info: [Subversion Indexer Paused with "Too many open files" Error](#).

7. Start FishEye!

In a terminal, change directory to `<FishEye install directory>` and run this:

```
bin/start.sh
```

After a few moments, in a web browser on the same machine, go to `http://localhost:8060/` (or, from another machine, type `http://hostname:8060/`, where hostname is the name of the machine where you installed FishEye).

Enter your license, then an admin password, to finish the setup. Note that this password is for the 'built-in' FishEye admin user. You can log in as this user, if necessary, by clicking the **Administration** link in the page footer.

You can postpone setting up JIRA integration until later if you wish; see [Configuring JIRA integration in the Setup Wizard](#).

8. Add repositories

Now you can tell FishEye about any existing repositories you have. Please read [Starting to use FishEye](#) for the details.

FishEye will perform an initial index of your repositories, during which it accesses, indexes and organizes a view of your repositories (including all historical items) back to the earliest commits. If you are evaluating FishEye, we suggest that you index a single project, so you can use FishEye as soon as possible. If you choose to index your entire repository, be aware that this can take a long time (possibly days) for massive or complex repositories and can be more complex to set up (especially for Subversion). The basic process is slightly different for each SCM type.

9. Add users and groups

You will want to set up your users and groups in FishEye. You can [add users directly](#) to FishEye, or connect to an [external user directory](#). Please read [Starting to use FishEye](#) for an introduction.

10. Set up your mail server

Configure the FishEye email server so that users can get notifications from FishEye. See [Configuring SMTP](#).

11. Connect to an external database (recommended)

If you intend to use this FishEye installation in a production environment, it is highly recommended that you use one of the [supported](#) external databases. See [Migrating to an external database](#).

If you are evaluating FishEye, or don't wish to do this now, FishEye will happily use its embedded HSQL database, and you can easily migrate later.

12. Stop FishEye (optional)

In a terminal, change directory to `<FishEye install directory>` and run this:

```
bin/stop.sh
```

13. Tuning FishEye performance

To get the best performance from your new FishEye installation, please consult [Tuning FishEye performance](#).

Starting to use FishEye

This page will guide you through the basics of using FishEye. By the end of it you should be able to:

- Create accounts for your collaborators, and organize them into groups.
- Add repositories that need to be indexed and setup permissions.
- Use the Commit Graph to trace the history of your code

This page assumes that:

- You have installed and started the latest version of FishEye. See [Installing FishEye on Linux and Mac](#) or [Installing FishEye on Windows](#) for details.
- You are using a [supported browser](#).

On this page:

- [Create users in FishEye](#)
- [Add a repository](#)
- [Move forward](#)

Related pages:

- [Installing FishEye on Windows](#)
- [Installing FishEye on Linux and Mac](#)
- [Supported platforms](#)
- [Managing your repositories](#)
- [Setting up your Users and Security](#)

Create users in FishEye

FishEye doesn't have any user accounts after you have installed it for the first time. You need to go to the Administration interface to add the first users of the system.

Click on the **Administration** link in the footer:

Atlassian FishEye analysis with Crucible code review. (Version:3.10.0 Build:r62dc31d 2015-08-07) - Administration - Page generated 2015-08-11 04:30 +0000

In the **Users** listing page click **Add user** to go to the user creation form:

Users ?

The User Browser allows you to browse all the users in the system. Filters allow you to limit the users that you see.

[Reset filter](#)

Filter In group Any Filter Add user

Fill in the form and create your user:

in

Settings
Users
Groups
Repositories
Schemes
URL Mappings
Customization
Links
Logs

Add New User

Information about the new user

Username

Display name

Email

Auth Type

Password

Confirm Password

From the **User** page you can click on **Create another user** to repeat this operation:

User

The user John Doe has been created.
[Create another user.](#)

User Details

Username **jdoe**

Display name **John Doe**

Email **john@doe.com**

Type **built-in**

Admin **false**

[Edit](#) | [Change Password](#) | [Rename](#) | [Delete](#)

Group Details

Groups **No groups.**

[Edit Groups](#)

List of users

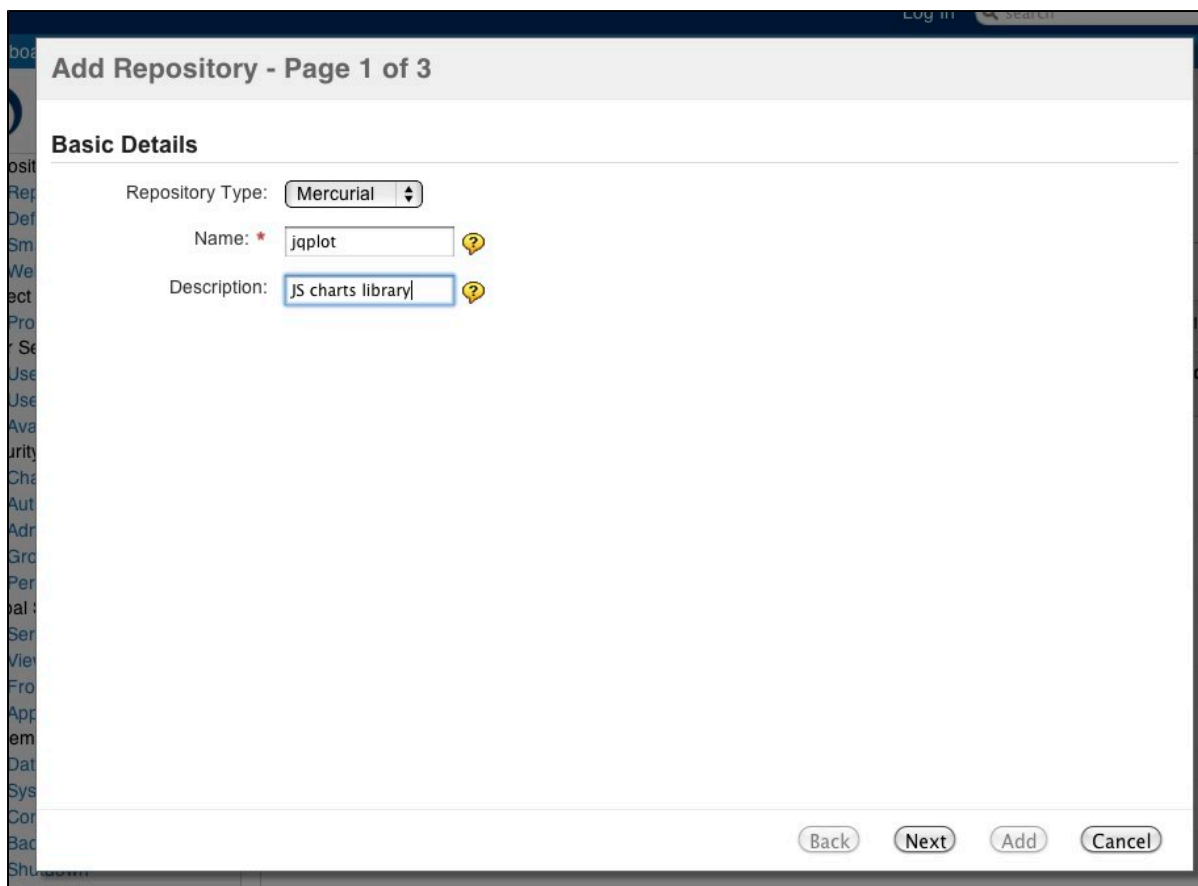
[Return to the list of users](#)

Add a repository

In this section we're going to add a repository to FishEye.

Click on **Add repository** in the **Repositories** listing of the Administration.

Now choose the repository type and fill in the name and description:



Add Repository - Page 1 of 3

Basic Details

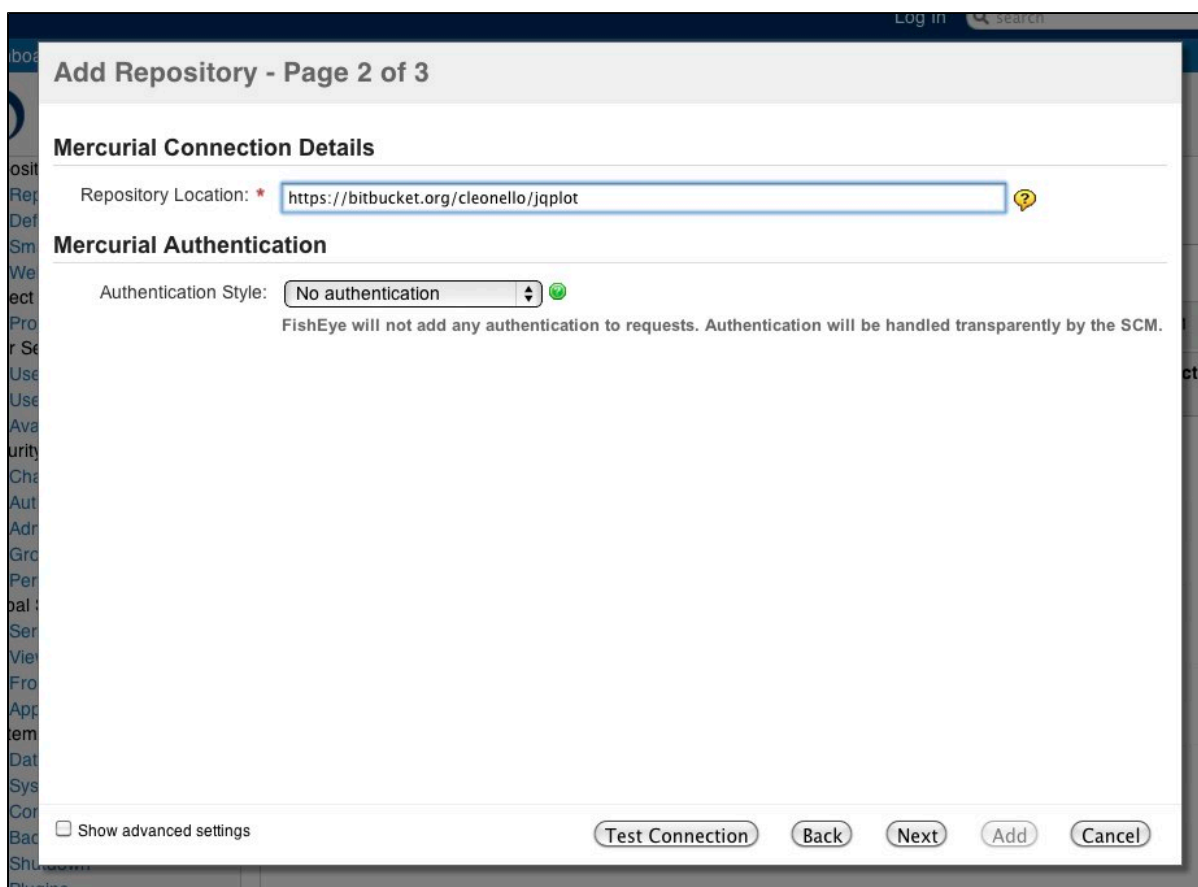
Repository Type: Mercurial

Name: * ?

Description: ?

Back Next Add Cancel

In the repository configuration, add the location of your repository. Fill in the authentication details if necessary.



Add Repository - Page 2 of 3

Mercurial Connection Details

Repository Location: * ?

Mercurial Authentication

Authentication Style: No authentication ✓

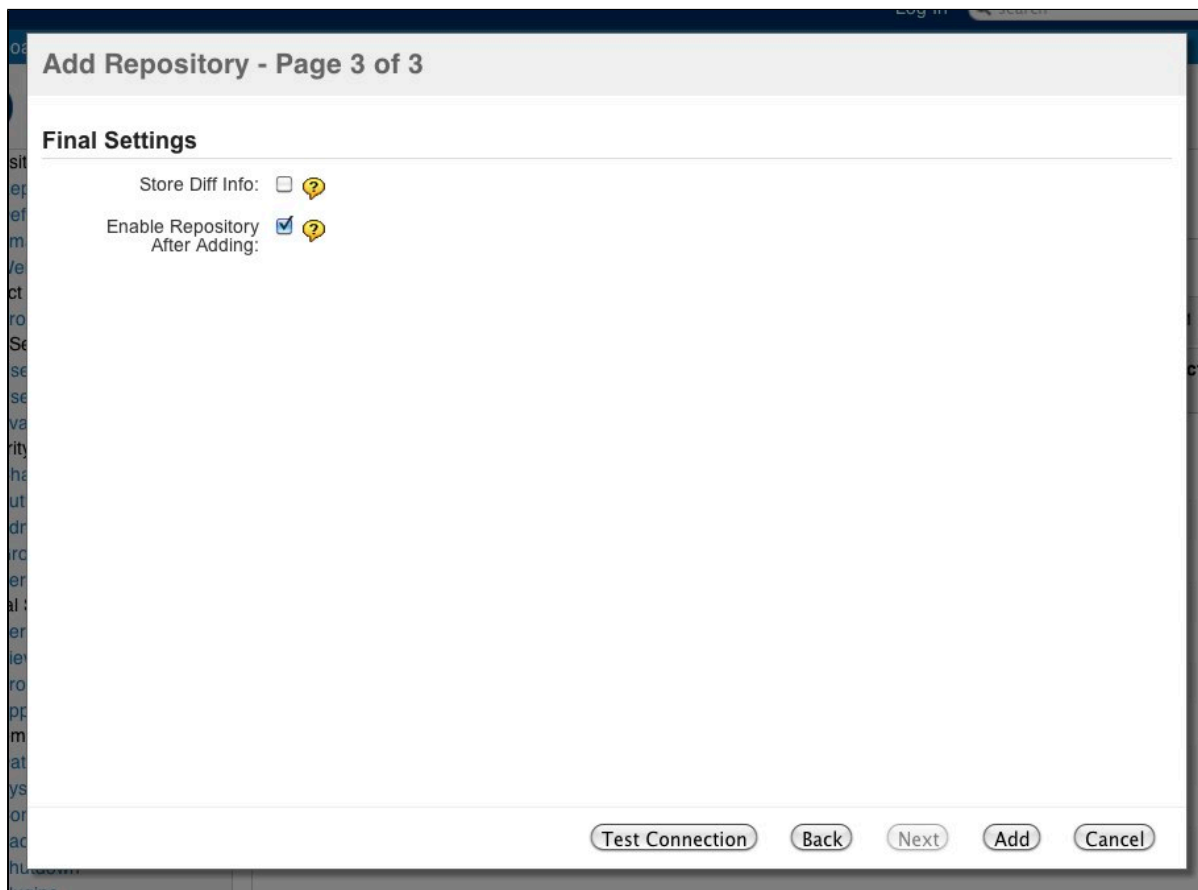
FishEye will not add any authentication to requests. Authentication will be handled transparently by the SCM.

☐ Show advanced settings

Test Connection Back Next Add Cancel

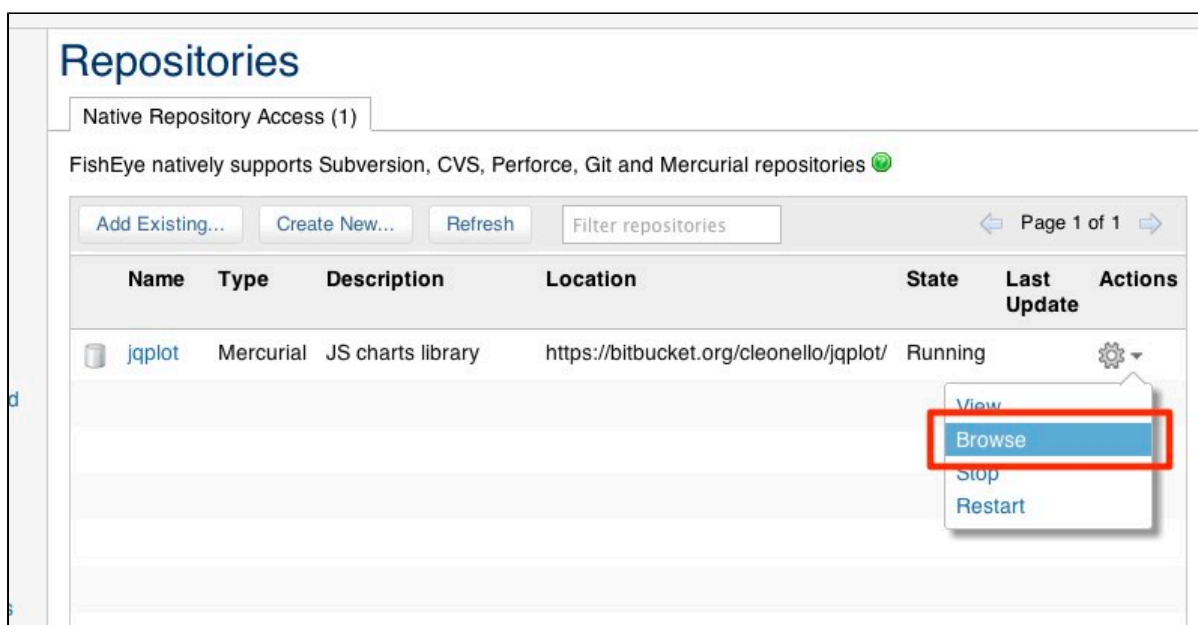
Finally indicate whether or not you would like diff indexing should be turned on and if the repository should not be indexed right away.

Click **Add** to finish the process.

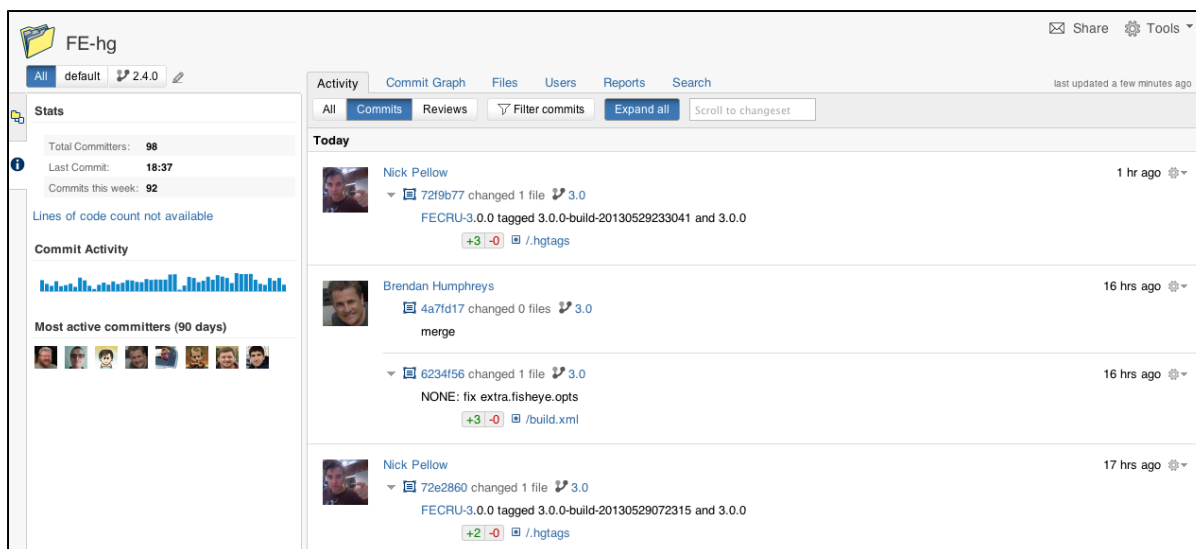


Move forward

Once it's created you can click **Browse**, in the repository options menu, to access your repository.



You can now browse your files in FishEye, search through your code or track modifications via the commit graph.



Configuring JIRA integration in the Setup Wizard

This page describes the **Connect to JIRA** tab of the FishEye setup wizard.

Connecting FishEye to a JIRA application allows you to manage your users with JIRA. See [Connecting to JIRA for user management](#) for more information.

On this page:

- [Connecting to a JIRA application in the Setup Wizard](#)
- [Troubleshooting](#)

Related pages:

- [Starting to use FishEye](#)
- [Linking FishEye to JIRA applications](#)
- [JIRA Integration in FishEye](#)

Connecting to a JIRA application in the Setup Wizard

To configure JIRA integration while running the FishEye setup wizard:

1. Configure the following setting in the JIRA application: [Allow remote API access](#).
2. Enter the following information on the 'Connect to JIRA' step of the setup wizard:

JIRA Base URL	The base URL set for your JIRA application. Examples are: <code>http://www.example.com:8080/jira/</code> <code>http://jira.example.com</code>
Admin Username	The credentials for a user with the 'JIRA System Administrators' global permission in the JIRA application.
Admin Password	
FishEye Base URL	Click Advanced Options to see this field. The JIRA application will use this URL to access your FishEye server. The URL you give here will override the base URL specified in your FishEye administration console, for the purposes of the JIRA connection.
Groups to Synchronize	Click Advanced Options to see this field. Select at least one JIRA application group to synchronize. The default group is <code>jira-users</code> . The JIRA application will synchronize all changes in the user information on a regular basis. The default synchronization interval is 1 hour.

Admin Groups

Click **Advanced Options** to see this field. Specify a JIRA group whose members should have administrative access to FishEye/Crucible. The default group is `jira-administrators`.

3. Click **Connect to JIRA**.
4. Finish the setup process.

Troubleshooting

▼ [Click to see troubleshooting information...](#)

This section describes the possible problems that may occur when integrating your application with JIRA via the setup wizard, and the solutions for each problem.

Symptom	Cause	Solution
The setup wizard displays one of the following error messages: - Failed to create application link from JIRA server at <URL> to this <application> server at <URL>. - Failed to create application link from this <application> server at <URL> to JIRA server at <URL>. - Failed to authenticate application link from JIRA server at <URL> to this <application> server at <URL>. - Failed to authenticate application link from <application> server at <URL> to this JIRA server at <URL>.	The setup wizard failed to complete registration of the peer-to-peer application link with JIRA. JIRA integration is only partially configured.	Remove the partial configuration if it exists, try the 'Connect to JIRA' step again, and then continue with the setup. Detailed instructions are below.

The setup wizard displays one of the following error messages: - Failed to register <application> configuration in JIRA for shared user management. Received invalid response from JIRA: <response> - Failed to register <application> configuration in JIRA for shared user management. Received: <response>	The setup wizard failed to complete registration of the client-server link with JIRA for user management. The peer-to-peer link was successfully created, but integration is only partially configured.	Remove the partial configuration if it exists, try the 'Connect to JIRA' step again, and then continue with the setup. Detailed instructions are below.
The setup wizard displays the following error message: Error setting Crowd authentication	The setup wizard successfully established the peer-to-peer link with JIRA, but could not persist the client-server link for user management in your config.xml file. This may be caused by a problem in your environment, such as a full disk.	Please investigate and fix the problem that prevented the application from saving the configuration file to disk. Then remove the partial configuration if it exists, try the 'Connect to JIRA' step again, and then continue with the setup. Detailed instructions are below.
The setup wizard displays the following error message: Error reloading Crowd authentication	The setup wizard has completed the integration of your application with JIRA, but is unable to start synchronizing the JIRA users with your application.	Restart your application. You should then be able to continue with the setup wizard. If this solution does not work, please contact Atlassian Support.
The setup wizard displays the following error message: An error occurred: java.lang.IllegalStateException: Could not create the application in JIRA/Crowd (code: 500). Please refer to the logs for details.	The setup wizard has not completed the integration of your application with JIRA. The links are only partially configured. The problem occurred because there is already a user management configuration in JIRA for this <application> URL.	Remove the partial configuration if it exists, try the 'Connect to JIRA' step again, and then continue with the setup. Detailed instructions are below.
No users can log in after you have set up the application with JIRA integration.	Possible causes: - There are no users in the group that you specified on the 'Connect to JIRA' screen. - For FishEye: There are no groups specified in the 'groups to synchronize' section of your administration console. - For Stash: You may not have granted any JIRA groups or users permissions to log in to Stash.	Go to JIRA and add some usernames to the group. - For FishEye: Go to the FishEye administration screens and specify at least one group to synchronize. The default is 'jira-users'. - For Stash: Grant the Stash User permission to the relevant JIRA groups on the Stash Global permissions page. If this solution does not work, please contact Atlassian Support.

Solution 1: Removing a Partial Configuration – The Easiest Way

If the application's setup wizard fails part-way through setting up the JIRA integration, you may need to remove the partial configuration from JIRA before continuing with your application setup. Please follow the

steps below.

Remove the partial configuration if it exists, try the 'Connect to JIRA' step again, and then continue with the setup wizard:

1. Log in to JIRA as a user with the '**JIRA System Administrators**' global permission.
2. Click the '**Administration**' link on the JIRA top navigation bar.
3. Remove the application link from JIRA, if it exists:
 - a. Click **Application Links** in the JIRA administration menu. The 'Configure Application Links' page will appear, showing the application links that have been set up.
 - b. Look for a link to your application. It will have a base URL of the application linked to JIRA. For example:
 - If you want to remove a link between JIRA and FishEye, look for the one where the **Application URL** matches the base URL of your FishEye server.
 - If you want to remove a link between JIRA and Confluence, look for the one where the **Application URL** matches the base URL of your Confluence server.
 - If you want to remove a link between JIRA and Stash, look for the one where the **Application URL** matches the base URL of your Stash server.
 - c. Click **Delete** next to the application link that you want to delete.
 - d. A confirmation screen will appear. Click **Confirm** to delete the application link.
4. Remove the user management configuration from JIRA, if it exists:
 - a. Go to the JIRA administration screen for configuring the applications that have been set up to use JIRA for user management:
 - In JIRA 4.3: Click '**Other Applications**' in the '**Users, Groups & Roles**' section of the JIRA administration screen.
 - In JIRA 4.4: Select '**Administration**' > '**Users**' > '**JIRA User Server**'.
 - b. Look for a link to your application. It will have a name matching this format:

```
<Type> - <HostName> - <Application ID>
```

For example:

```
FishEye / Crucible - localhost -  
92004b08-5657-3048-b5dc-f886e662ba15
```

Or:

```
Confluence - localhost -  
92004b08-5657-3048-b5dc-f886e662ba15
```

If you have multiple servers of the same type running on the same host, you will need to match the application ID of your application with the one shown in JIRA. To find the application ID:

- Go to the following URL in your browser:

```
<baseUrl>/rest/applinks/1.0/manifest
```

Replace <baseUrl> with the base URL of your application.

For example:

```
http://localhost:8060/rest/applinks/1.0/manifest
```

- The application links manifest will appear. Check the application ID in the <id> element.
 - c. In JIRA, click '**Delete**' next to the application that you want to remove.
5. Go back to the setup wizard and try the 'Connect to JIRA' step again.

Solution 2: Removing a Partial Configuration – The Longer Way

If solution 1 above does not work, you may need to remove the partial configuration and then add the full integration manually. Please follow these steps:

1. Skip the 'Connect to JIRA' step and continue with the setup wizard, to complete the initial configuration of the application.
2. Log in to JIRA as a user with the '**JIRA System Administrators**' global permission.
3. Click the '**Administration**' link on the JIRA top navigation bar.
4. Remove the application link from JIRA, if it exists:
 - a. Click **Application Links** in the JIRA administration menu. The 'Configure Application Links' page will appear, showing the application links that have been set up.
 - b. Look for a link to your application. It will have a base URL of the application linked to JIRA. For example:
 - If you want to remove a link between JIRA and FishEye, look for the one where the **Application URL** matches the base URL of your FishEye server.
 - If you want to remove a link between JIRA and Confluence, look for the one where the **Application URL** matches the base URL of your Confluence server.
 - If you want to remove a link between JIRA and Stash, look for the one where the **Application URL** matches the base URL of your Stash server.
 - c. Click **Delete** next to the application link that you want to delete.
 - d. A confirmation screen will appear. Click **Confirm** to delete the application link.
5. Remove the user management configuration from JIRA, if it exists:
 - a. Go to the JIRA administration screen for configuring the applications that have been set up to use JIRA for user management:
 - In JIRA 4.3: Click '**Other Applications**' in the '**Users, Groups & Roles**' section of the JIRA administration screen.
 - In JIRA 4.4: Select '**Administration**' > '**Users**' > '**JIRA User Server**'.
 - b. Look for a link to your application. It will have a name matching this format:

```
<Type> - <HostName> - <Application ID>
```

For example:

```
FishEye / Crucible - localhost -  
92004b08-5657-3048-b5dc-f886e662ba15
```

Or:

```
Confluence - localhost -  
92004b08-5657-3048-b5dc-f886e662ba15
```

If you have multiple servers of the same type running on the same host, you will need to match the application ID of your application with the one shown in JIRA. To find the application ID:

- Go to the following URL in your browser:

```
<baseUrl>/rest/applinks/1.0/manifest
```

Replace <baseUrl> with the base URL of your application.

For example:

```
http://localhost:8060/rest/applinks/1.0/manifest
```

- The application links manifest will appear. Check the application ID in the <id> element.
- c. In JIRA, click '**Delete**' next to the application that you want to remove.

6. Add the application link in JIRA again, so that you now have a two-way trusted link between JIRA and your application:
 - a. Click **Add Application Link**. Step 1 of the link wizard will appear.
 - b. Enter the **server URL** of the application that you want to link to (the 'remote application').
 - c. Click **Next**.
 - d. Enter the following information:
 - **Create a link back to this server** – Check to add a two-way link between the two applications.
 - **Username and Password** – Enter the credentials for a username that has administrator access to the remote application.
Note: These credentials are only used to authenticate you to the remote application, so that Application Links can make the changes required for the new link. The credentials are not saved.
 - **Reciprocal Link URL** – The URL you give here will override the base URL specified in your remote application's administration console, for the purposes of the application links connection. Application Links will use this URL to access the remote application.
 - e. Click **Next**.
 - f. Enter the information required to configure authentication for your application link:
 - **The servers have the same set of users** – Check this box, because the users are the same in both applications.
 - **These servers fully trust each other** – Check this box, because you trust the code in both applications and are sure both applications will maintain the security of their private keys.
For more information about configuring authentication, see [Configuring authentication for an application link](#).
 - g. Click **Create**.
7. Configure a new connection for user management in JIRA:
 - a. Go to the JIRA administration screen for configuring the applications that have been set up to use JIRA for user management:
 - In JIRA 4.3: Click '**Other Applications**' in the '**Users, Groups & Roles**' section of the JIRA administration screen.
 - In JIRA 4.4: Select '**Administration**' > '**Users**' > '**JIRA User Server**'.
 - b. **Add** an application.
 - c. Enter the **application name** and **password** that your application will use when accessing JIRA.
 - d. Enter the **IP address** or addresses of your application. Valid values are:
 - A full IP address, e.g. 192.168.10.12.
 - A wildcard IP range, using CIDR notation, e.g. 192.168.10.1/16. For more information, see the introduction to [CIDR notation on Wikipedia](#) and [RFC 4632](#).
 - **Save** the new application.
8. Set up the JIRA user directory in the application.
 - For Confluence:
 - a. Go to the **Confluence Administration Console**.
 - b. Click '**User Directories**' in the left-hand panel.
 - c. **Add** a directory and select type '**Atlassian JIRA**'.
 - d. Enter the following information:
 - **Name** – Enter the name of your JIRA server.
 - **Server URL** – Enter web address of your JIRA server. Examples:


```
http://www.example.com:8080/jira/
http://jira.example.com
```
 - **Application name and Application password** – Enter the values that you defined for Confluence in the settings on JIRA.
 - e. Save the directory settings.
 - f. Define the **directory order** by clicking the blue up- and down-arrows next to each directory on the '**User Directories**' screen.
For details see [Connecting to Crowd or JIRA for User Management](#).
 - For FishEye/Crucible:
 - a. Click **Authentication** (under 'Security Settings').
 - b. Click **Setup JIRA/Crowd authentication**. Note, if LDAP authentication has already been

set up, you will need to remove that before connecting to JIRA for user management.

c. Make the following settings:

Authenticate against	Select a JIRA instance
Application name and password	Enter the values that you defined for your application in the settings on JIRA.
JIRA URL	The web address of your JIRA server. Examples: <code>http://www.example.com:8080/jira/</code> <code>http://jira.example.com</code>
Auto-add	Select Create a FishEye user on successful login so that your JIRA users will be automatically added as a FishEye user when they first log in.
Periodically synchronise users with JIRA	Select Yes to ensure that JIRA will synchronize all changes in the user information on a regular basis. Change the value for Synchronise Period if required.
When Synchronisation Happens	Select an option depending on whether you want to allow changes to user attributes from within FishEye.
Single Sign On	Select Disabled . SSO is not available when using JIRA for user management and if enabled will make the integration fail.

d. Click **Next** and select at least one user group to be synchronised from JIRA. If necessary, you could create a new group in JIRA, such as 'fisheye-users', and select this group here.

e. Click **Save**.

- For Stash:

- a. Go to the Stash administration area.

- b. Click **User Directories** in the left-hand panel.

- c. **Add** a directory and select type **Atlassian JIRA**.

- d. Enter the following information:

- **Name** – Enter the name of your JIRA server.

- **Server URL** – Enter web address of your JIRA server. Examples:

`http://www.example.com:8080/jira/`
`http://jira.example.com`

- **Application name** and **Application password** – Enter the values that you defined for Stash in the settings on JIRA.

- e. Save the directory settings.

- f. Define the directory order by clicking the blue up- and down-arrows next to each directory on the 'User Directories' screen.

For details see [Connecting Stash to JIRA for user management](#).

Using FishEye

- [Using the FishEye screens](#)
 - [Browsing through a repository](#)

- Viewing file content
- Viewing a file history
- Using side by side diff view
- Viewing the changelog
- Searching FishEye
- FishEye charts
- Using favourites in FishEye
- Changeset discussions
- Viewing the commit graph for a repository
- Viewing people's statistics
- Using Smart Commits
- Changing your user profile
 - Re-setting your password
- Pattern matching guide
- Date expressions reference guide
- EyeQL reference guide
- Integrating FishEye with Atlassian applications
 - JIRA Integration in FishEye
 - Integrating FishEye with Bitbucket Server
- Transitioning issues in JIRA

Using the FishEye screens

The sections below describe the different screens in FishEye and the information you can retrieve from them. Each page (tab) has a number of panes, each of which is described separately below.

Header



The header along the top of each FishEye page provides the following:

- The application navigator, at the left of the header, connects you directly to your other applications, such as JIRA and Bamboo. Admins can configure which apps appear in the navigator – just click **Application navigator** in the admin area.
- FishEye logo (with the logo for Crucible if you are using that) – click to go to the dashboard, to see your personal code commits, your reviews (if you are using [Crucible](#)) and your activity stream.
- **Repositories** — the list of all FishEye repositories. Click a repository name to [browse the repository](#). A number of sub-tabs become available, as described below (see 'Repository sub-tabs' below).
- **Projects** (*when used with Crucible*) – a link to all projects (see the [Crucible documentation](#)). Logged-in users can see links to recently visited projects.
- **People** – tab to view statistics about committers to your FishEye repositories (see [Viewing People's Statistics](#)). Logged in users can see links to users they have recently visited.
- **Reviews** (*when used with Crucible*) – go to your code reviews (see the [Crucible documentation](#)). Logged-in users can see links to recently visited reviews as well as to the Crucible 'Inbox' and 'Outbox'.
- **Create review** (*when used with Crucible*) – click the down arrow to choose **Create snippet**.
- **+** – add a new repository. See [Adding an external repository](#).
- **Search** – use FishEye's powerful search engine to find changesets, committers and files. See [Searching FishEye](#).
- Click your avatar to change your user settings (see [Changing your user profile](#)).

Repository tabs

Once you have selected a repository, you can navigate through it by selecting files and folders in the navigation tree on the left. The available tabs change according to whether you are viewing a repository or a file:

Tab	Repo	File	Description

Files	✓		Provides details about the files in the repository. See Browsing through a repository .
Revisions		✓	Shows the latest revisions of the file. See Viewing a file history .
Activity	✓	✓	Shows recent activity on the item. There are a number of sub-options here (see Viewing the changelog): • All — The default view, showing commits, reviews (when used with Crucible) and JIRA issues (when used with JIRA). • Commits — Shows commits in the activity stream. • Reviews — Shows review activity in the activity stream (when used with Crucible). • Filter — Applies constraints to the current activity stream. • Scroll to Changeset — Opens the changeset ID specified in the text field (press Enter to carry out the action).
Commit graph	✓		Provides a visual representation of commits to, and branches in, the repository. See Viewing the commit graph for a repository .
Users	✓	✓	Shows the commit history of the different users that have committed changes on the item.
Reports	✓	✓	Shows activity charts for the item. Various chart options can be selected in the left navigation bar (see FishEye charts).
Source		✓	Shows the contents of the file. See Viewing file content .
Search	✓		Gives access to the advanced search capabilities in FishEye.

Browsing through a repository

Browse your repositories in FishEye to see information about the files in the repository and related activity, including commits to the repository. You can also generate charts, and search for specific file revisions in the repository.

On this page:

- [Browse a repository](#)
- [View information about a repository or directory](#)
- [View information about a file](#)
- [Hide empty directories and deleted files](#)
- [Watch a repository](#)

Browse a repository

1. Click **Repositories** in the header and choose either a recently viewed repository, or **All repositories**.
2. Click the name of a repository to view its contents.
3. If required, use the branch/tag selector (just above the file tree) to change context.

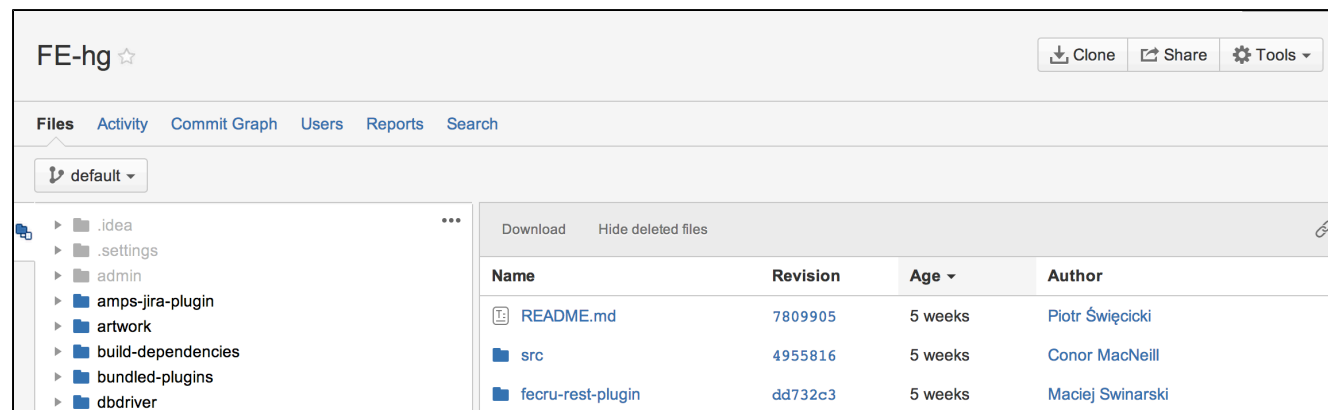
View information about a repository or directory

Click a folder/directory name to see information about that on these tabs:

Files	View the contents of the repository/folder.
Activity	View the commit, review and issue (requires JIRA) activity related to the repository/folder. The activity stream is similar to the changelog activity stream. See Viewing the changelog .
Commit Graph	Visualize the repository, using the commit graph. See Viewing the commit graph for a repository .

Users	View the commit history of the users that have committed changes to files in the repository/folder. See Viewing People's Statistics .
Reports	View activity charts for the repository/folder. See FishEye charts .
Search	Search the repository/folder. See Searching FishEye .

 A greyed out item is either deleted or empty.



FE-hg ☆

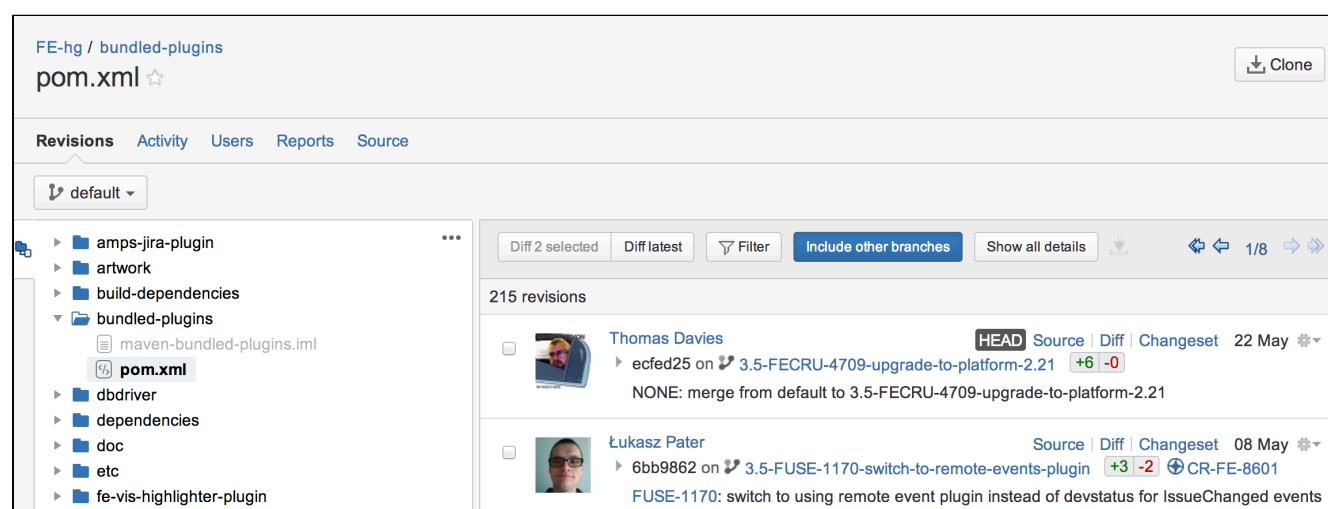
Download Hide deleted files

Name	Revision	Age	Author
README.md	7809905	5 weeks	Piotr Święcicki
src	4955816	5 weeks	Conor MacNeill
fecru-rest-plugin	dd732c3	5 weeks	Maciej Swinarski

View information about a file

Click a file name to see information about the file on these tabs:

Revisions	The history of revisions for the file. See Viewing a file history .
Activity	The commits and reviews activity related to the file. See Viewing the changelog .
Users	Commit histories for users who have committed changes to the file. See Viewing people's statistics .
Reports	Charts for the file activity. See FishEye charts .
Source	The raw file can be downloaded from this tab. See Viewing file content .



FE-hg / bundled-plugins

pom.xml ☆

Revisions Activity Users Reports Source

Diff 2 selected Diff latest Filter Include other branches Show all details

215 revisions

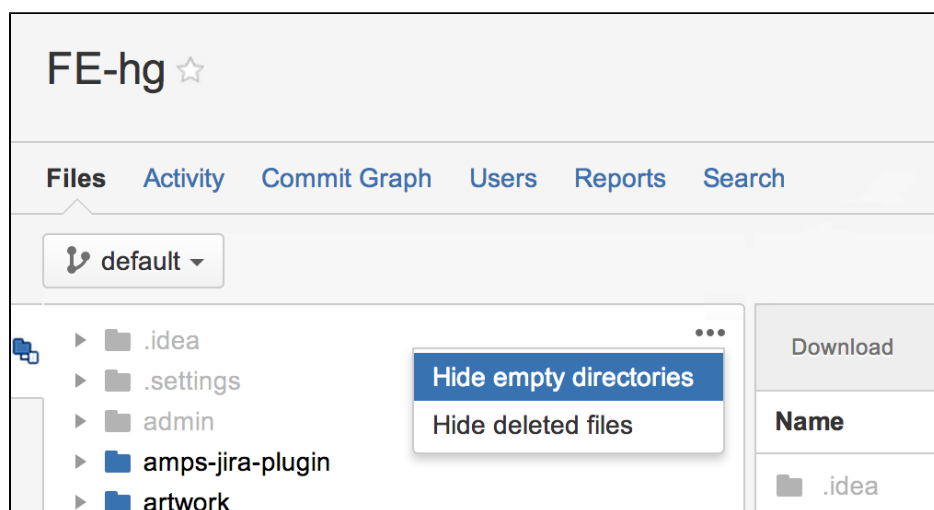
	Thomas Davies	HEAD	Source	Diff	Changeset	22 May
ecfed25 on 3.5-FECRU-4709-upgrade-to-platform-2.21 +6 -0						
NONE: merge from default to 3.5-FECRU-4709-upgrade-to-platform-2.21						
	Łukasz Pater		Source	Diff	Changeset	08 May
6bb9862 on 3.5-FUSE-1170-switch-to-remote-events-plugin +3 -2 CR-FE-8601						
FUSE-1170: switch to using remote event plugin instead of devstatus for IssueChanged events						

Hide empty directories and deleted files

FishEye tracks deleted files for your repository. Deleted files will be greyed out in the left-hand navigation tree. If all of the files in a directory are deleted, the directory will also be greyed out.

You can hide deleted files and empty directories in the left navigation tree:

1. Click **Repositories** in the header and browse to a repository.
2. In the left hand navigation panel, click the 'Actions' menu and choose either:
 - **Hide empty directories** – hide all empty (greyed out) directories and their contents (i.e. deleted files and empty sub-directories).
 - **Hide deleted files** – hide all deleted (i.e. greyed out) files. This does not affect directories.



i If you hide both empty directories and deleted files, you will only see files and directories that exist on the [Head](#) of that path. In repositories other than Subversion repositories, this could mean files/directories on any branch.

Watch a repository

You can "watch" a repository in FishEye/Crucible. Watching the repository allows you to receive email notifications when changes are made to the repository.

1. Navigate to the repository that you want to watch.
2. Choose **Tools > Watch**.

You can view all of your watches and configure the frequency of your watch emails in your user profile. See [Changing your user profile](#).

To remove the watch, choose **Tools > Unwatch**. You can also remove watches from within your user profile.

The option to add a watch will only be available if the administrator has [enabled watches](#) for the repository.

Viewing file content

You can [search](#) or [browse](#) your repositories in FishEye to view a specific file. Once you find the file, you can view the diffs between different revisions of the file, and see any annotations.

You can also download the source code for the file.

View the source code for a file

1. [Search](#), or [browse](#) through a repository, to find the file.
2. Click the file name to see the revision history for the file.
3. On the **Source** tab:

Displaying a diff	Click the arrow buttons to see the previous or next revisions of the file. Pick revision numbers (e.g. 'b3ebfaf') from the two lists to display the diff for those two revisions.
Changeset	View the changeset that the revision was a part of.
Raw	Download the raw source code for the file.

Reviews	Select Create Review to create a Crucible review from the file, or go to a review that relates to the file. <i>(Requires Crucible)</i>
Blame	Press to display authors and revision numbers next to each line of code in the file

Viewing a file history

You can view a specific file when [browsing a repository](#). This allows you to see information about the file, including the history of file revisions.

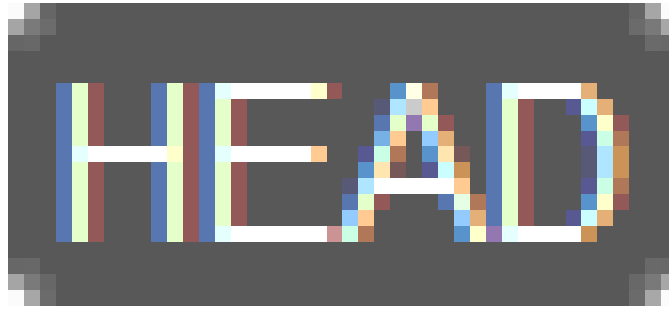
View the history of revisions for a file

1. Log into FishEye/Crucible.
2. [Search](#), or [browse](#) through a repository, to find the file.
3. Click the **Revisions** tab. The history of revisions for the file will be displayed. See the '[File Revisions](#)' screenshot below.

Diff 2 selected	Check boxes for two file revisions and then click to view the diff for those revisions.
Diff latest	View the diff of the two most recent file revisions.
Filter	View the file filter. Enter the desired fields to filter the file history results on.
Include other branches	Include revisions of the same file from other branches. A file can have many physical paths, all of which relate to the same filename in your project structure, or repository's logical structure. This applies to Subversion's branching and tagging directory structure in particular.

Show all details

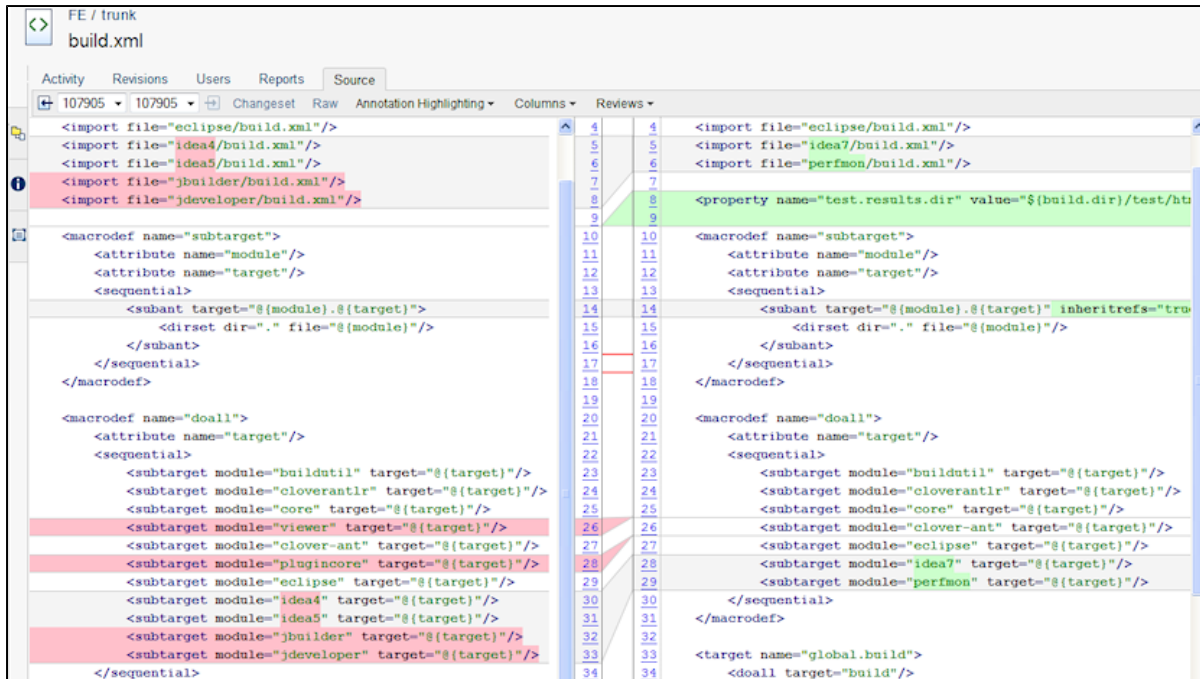
Toggle expand or collapse of all file revisions to show additional information including the revision ID, parents and the branch where it is head, denoted with this graphic:



See the '[Overview of a File Revision](#)' diagram below for an explanation of the information provided about individual revisions.

Screenshot: File Revisions
Diagram: Overview of a file revision
Using side by side diff view

FishEye's 'side-by-side diff' view allows you to see how a file's content has changed. Different versions of the file are displayed together to make line-by-line comparisons easy.



Show the side-by-side diff view

1. Open the source code view for the file. See [Viewing file content](#).
2. Use the elect a range of revisions to compare.
3. Choose **View > Side-by-side Diff**.

Note that:

- Green highlighting indicates where lines have been added; red shows where they've been removed.
- Grey highlights indicate that a line's internal content has changed.
- Each addition or deletion is linked to the adjacent pane by a colored triangle to show the location of that change in the comparison file.
- Line numbers in the margin are permanent links ("permalinks") that can be sent to colleagues. When they open those links, the view will automatically open in side by side diff mode.

Alternative ways to open side-by-side diffs

From the FishEye Dashboard

You can also open side by side diffs from the Dashboard screens, by clicking the 'Delta' triangle icon next to an item when it appears in the stream of events. This will open the file in the diff view. If you have currently selected side by side diff as the viewing mode, then the diff will automatically be displayed in that mode. If not, you can select side by side diff from the **View** menu.

From the Revisions History view

When in the revisions view, you can show a diff by checking the boxes next to two revisions, then clicking the **Diff** in the top control bar. If you have currently selected side by side diff as the viewing mode, then the diff will automatically be displayed in that mode. If not, you can select side by side diff from the **View** menu.

You can also launch into a diff of the latest revision and the second most recent by clicking **Latest Diff** in the top control bar.

Viewing the changelog

The changelog is a record of the commits and reviews for a repository, branch, directory or file. You can view the recent activity in the changelog when [browsing a repository/branch/directory](#).

On this page:

- [View changelog activity](#)
- [Filter commit activity in the changelog](#)
- [Watch the changelog activity](#)

View changelog activity

1. [Browse to the desired repository, branch, or directory.](#)
2. Use the branch/tag picker (under the repository or file name) to change context, if required.
3. Click the **Activity** tab.
4. Use the following functions of the **Activity** tab to see different aspects of the changelog activity:

All	Click to show commits, reviews (requires integration with Crucible) and JIRA issues (requires integration with JIRA) activity in the activity stream.
Commits	Click to show only commits in the activity stream.
Reviews	Show only review activity. (Requires integration with Crucible)
Filter commits	See Filter commit activity in the changelog (below) for more information.
Expand all	Show all modified files related to each changeset.
Scroll to changeset	Enter a changeset ID (e.g. 107856) to scroll to that changeset in the activity stream.

The screenshot shows the FishEye web interface for the 'FE-hg' repository. The 'Activity' tab is selected, displaying a list of commits. The interface includes a sidebar with a file tree, a top navigation bar with tabs like 'Files', 'Activity', 'Commit Graph', 'Users', 'Reports', and 'Search'. The 'Activity' tab is active, showing a list of commits. The commits are filtered by 'All' and 'Commits' tabs. The activity stream shows commits from Wednesday 04 Jun and Tuesday 03 Jun. Each commit entry includes a user profile picture, name, commit ID, and a brief description of the changes.

Filter commit activity in the changelog

You can filter the commits that are displayed in the activity streams on the **All** and **Commits** tabs of the **Activity** tab. Note that you cannot use the commits filter to filter reviews.

1. Go to the **Activity** tab, as described above.
2. When viewing either the **All** or **Commits** tab, click **Filter commits**.
3. Enter filtering criteria:

Committer	Changesets checked in by the given committer/author.
Log Comment	Changesets where the commit comment matches the given text.

File Extension	Changesets that contain files with the specified file extension.
File Name	Changesets that contain a given file.
Start Date	Changesets created on or after that date. Must be of the form YYYY-MM-DDTHH:mm:ss, YYYY-MM-DD, YYYY-MM or YYYY (you can use '/' instead of '-').
End Date	Changesets created on or before that date. Must be of the form YYYY-MM-DDTHH:mm:ss, YYYY-MM-DD, YYYY-MM or YYYY (you can use '/' instead of '-').

4. Click **Apply**.

- Use **Clear** to clear the filter.
- Click **Filter commits** again to turn off the filter.

Watch the changelog activity

You can "watch" a changelog's activity stream in FishEye and Crucible. Watching the activity stream means that you receive emails when updates occur in the activity stream. You can even add filtering criteria for the watch, so for example you can restrict watches for commits by a particular user.

Note, the option to add a watch will only be available if the administrator has [enabled watches](#) for the repository.

1. Navigate to the activity stream that you want to watch, as described above.
2. If required, add filtering criteria, as described in the [Filter commit activity in the changelog](#) section above, and click **Apply**.
3. Choose **Tools > Watch**.

To remove the watch from the activity stream, choose **Tools > Unwatch**. The watch will be removed.

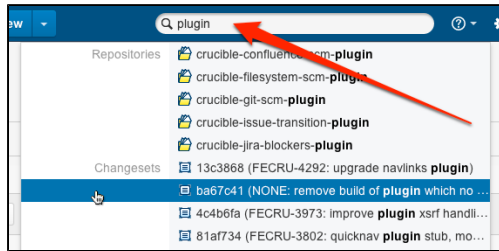
You can also view all of your watches, configure the frequency of your watch emails, and remove watches, in your user profile. See [Changing your user profile](#) for more information.

Searching FishEye

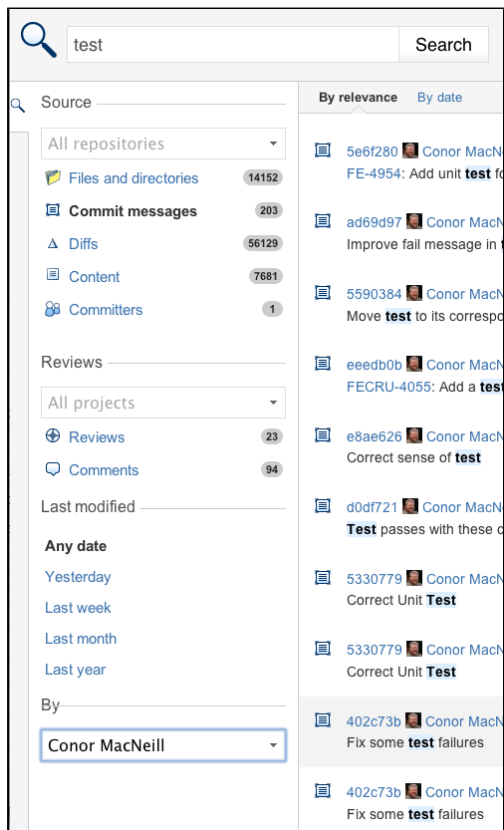
FishEye has a powerful search engine that allows you to find changesets, committers and files. There are two methods for searching in FishEye:

Quick Search

- Searches across all repositories connected to FishEye.
- Suggests "quick nav" results as you type.



- Has extra search tools, such as 'handles'.
- Press <Enter> to see filters:



Advanced Search

- Click the **Search** tab when viewing a repository.
- Searches a **single repository**.
- Access the **EyeQL** query language, if required.
- Enter search criteria for a range of attributes:

Search Criteria

☐ Search HEAD only

Comments:

Contents:

Added Text:

Removed Text:

Author(s):

File name:

(Ant-globs):

Branch:

Tag:

After:

Before:

(YYYY-MM-DD):

Order by:

Group by:

Display data:

☒ Path ☒ Revision ☒ Author ☒ Date

☒ Changeset ID ☐ Comment ☐ Total Lines

☐ Lines Added ☐ Lines Removed ☐ Tags

☐ Reviews

[Switch to EyeQL Search](#)

Use Quick Search

Simply enter your search criteria in the search box in the FishEye header to search across file, repository, committer and user names, as well as commit messages and reviews (if you are using **Crucible** with FishEye).

- Use search tools and filters to refine your results. See [Refining your Quick Search Criteria](#) and [Filtering Quick Search Results](#) below.
- Results are weighted by edit date; files edited within the last twelve months are given greater weighting.
- There is a 100-repository limit on searches, to prevent FishEye becoming unresponsive when there

are large numbers of repositories. FishEye will also limit the search to the specific repository that you are looking at, if you are already navigating within a specific repository

- If FishEye is integrated with JIRA you can see a summary of a JIRA issue in your search results by hovering over the issue key. See [JIRA Integration in FishEye](#).
- See also [Searching Crucible](#) in the Crucible documentation.

Refine your Quick Search criteria

The FishEye Quick Search has a number of powerful tools that you can use to refine your search criteria before executing the search.

Tool	Description	Example
CamelCase pattern matching	Enter a CamelCase pattern to find matching files and directories.	BooQCTest returns results like BooleanQueryCoordTest and BooleanQueryClassTest.
Path/File pattern matching	Enter a path/file pattern to find matching files and directories.	common/final/Actions returns results like /src/common/eu/systems/specialprojects/final/Actions.java.
Field handles	Use a field handle to restrict your search to a particular field: • file — file/directory names • commit — commit messages • diff — lines added/removed • content — file contents • committer — committer names	file:build.xml returns files with names matching build.xml.
Search within a directory	Use AntGlobs to search a specific directory.	Search for /src/**/gwt/*.xml and FishEye will return all files with a .xml extension that are below both a src and a gwt directory. e.g. src/java/com/atlassian/fecru/gwt/FecruCore.gwt.xml but not src/java/com/atlassian/fecru/ApplicationContext.xml
Search for discrete phrases	Enter a phrase in quotation marks. Not case-sensitive.	"update version in build" returns matches for the exact string, i.e. return "update build version" or "update version in file".

Filter Quick Search results

Once you have a set of search results on the Quick Search page, you can filter them to a subset of the original results. The filter controls are in the left panel of the Quick Search page in the 'Source' section.

Filter	Description
All repositories	By default, searches across all repositories. Choose a repo to restrict the search to just that one. For example, if you enter 'FE' then the search results page will refresh to display only files and directories in the 'FE' repository.
Files and directories	Filter your results to files and directories that have names that match the search criteria, see details below .
Commit messages	Filter your results to changeset comments that match the search criteria.
Diffs	Filter your results to diffs (lines added/removed) that match the search criteria.
Content	Filter your results to files that have content that match the search criteria, see details below .
Committers	Filter your results to committers that match the search criteria.
All projects	When using Crucible with FishEye, there is a 'Reviews' section. The All projects dropdown allows you to filter reviews returned in the search results. See Searching Crucible .
Reviews	Search in reviews
Comments	Search for review comments
Last modified	Filter by the date of the last change.
Author	Filter by author name.

Use Advanced Search

The Advanced Search can only be run against a specific repository, however you can specify more precise criteria against a number of fields for that repository.

1. Navigate to the repository that you want to search, as described in [Browsing through a repository](#).
2. Click the **Search** tab.
3. Enter your search criteria:
 - **Standard Search** — Enter criteria in the 'Search Criteria' panel. See [Specify criteria using the search interface](#) below for details.
 - **EyeQL Search** — Enter your "EyeQL" query. See [Specify criteria using EyeQL](#) section below for more details.

Use **Switch to EyeQL Search** and **Switch to Standard Search** at the bottom of the 'Search Criteria' panel to toggle between the two search methods.

Specify criteria using the search interface

The Advanced Search interface allows you to specify search criteria for multiple fields, order the results, group the results and choose the display fields in the results. You can link to the search results, as well as save the results to a CSV file.

Please note the following:

- **Contents** — Files must be non-binary, less than 5MB, and for SVN repositories, on trunk only. Only HEAD/tip revisions are searched. For older revisions, use the added/removed text search criteria.
- **File names** — [Antglobs](#) can be used to specify the criteria for this field.

Specify criteria using EyeQL

Advanced Search also allows you to run searches using FishEye's powerful EyeQL query language.

Click **Switch to EyeQL Search** at the bottom of the 'Search Criteria' panel.

See the [EyeQL reference guide](#) for more information. If you're unfamiliar with EyeQL, consider using the [Standard Search Interface](#) to build your initial query, then switch to EyeQL to modify that.

FishEye charts

When [browsing a repository](#), the **Reports** tab displays a chart of the lines of code (LoC) committed to the repository over time. The following options are available:

- [Charts](#)
- [Code Metrics](#)
- [Notes](#)

Charts

Click **Reports** and then **Charts** when browsing a repository to see charts of activity in the repository.

You can control the chart type that is displayed and various chart options. Click the cog icon on the left, select the required options, and click **Apply**. The available options include:

Setting	Explanation	Values	Default
Branch/Tag selector	Limits the chart to the selected branch/tag.	Any branch/tag from the current repository.	Displays the default/trunk.
Chart type	Changes the chart's presentation.	Area, line, pie or change* chart.	Area
Author	Limits the chart to show specific author(s).	Any author configured in the system.	All
Extension	Limits the chart to show specific file type(s).	Any file extension; e.g. '.java'.	All
Subdirectory	Limit the chart to a folder under the current branch. Files in the current directory are represented by an element labeled '.(this dir)'.	A single folder.	None (show all)
Start Date	Date of the earliest data to show.	Date in format YYYY-MM-DD.	None (show all)
End Date	Date of the latest data to show.	Date in format YYYY-MM-DD.	None (show all)

 *The 'Change' chart displays the change in lines of code, for a specific date range, expressed as a line graph. For example, if the lines of code at the start date is 100, the start point will be zero and the rest of the graph shifted by 100 lines.

Screenshot: FishEye custom chart settings

All **trunk**

Chart types:

☐ Area ☒ Line ☐ Pie ☐ Change

Chart options:

Break down by:

☐ None ☒ Subdirectory ☐ Author ☐ User ☐ Extension

Date Range:

Start Date:

:

Author(s):

(any)

Adam Ahmed (aahmed)

Anna Buttfield (abuttfield)

Ben Speakmon (bspeakmon)

Brendan Humphreys (bhumphreys)

File Extension:

(any)

None

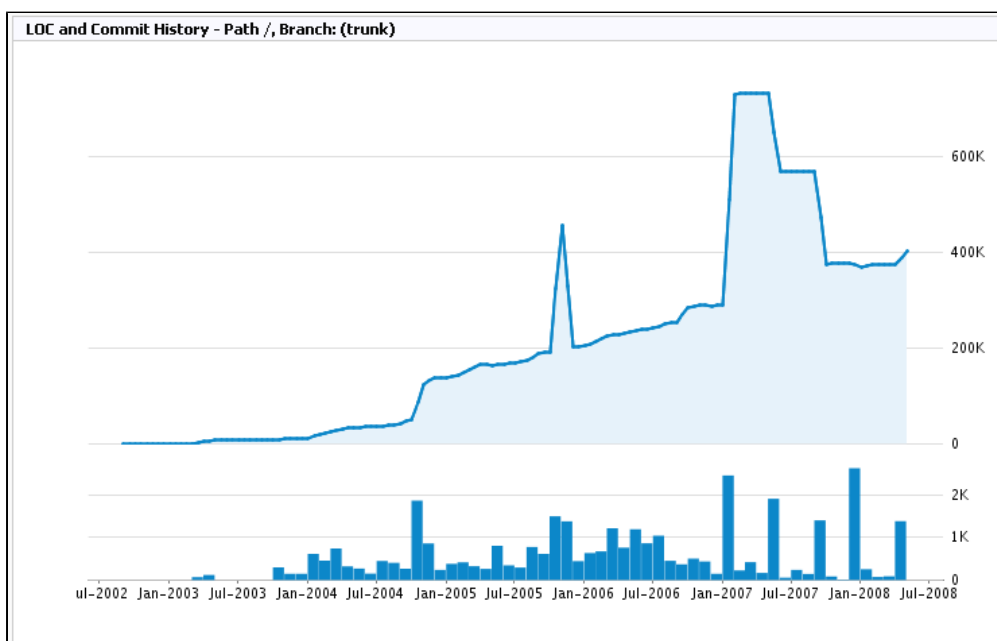
.MF

.TXT

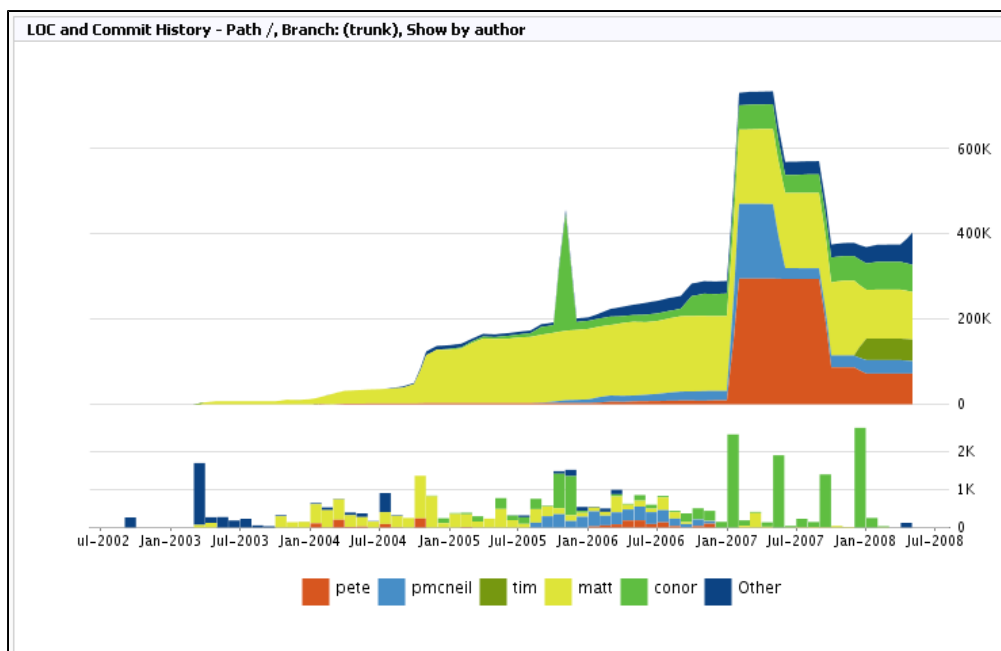
.cfs

Apply

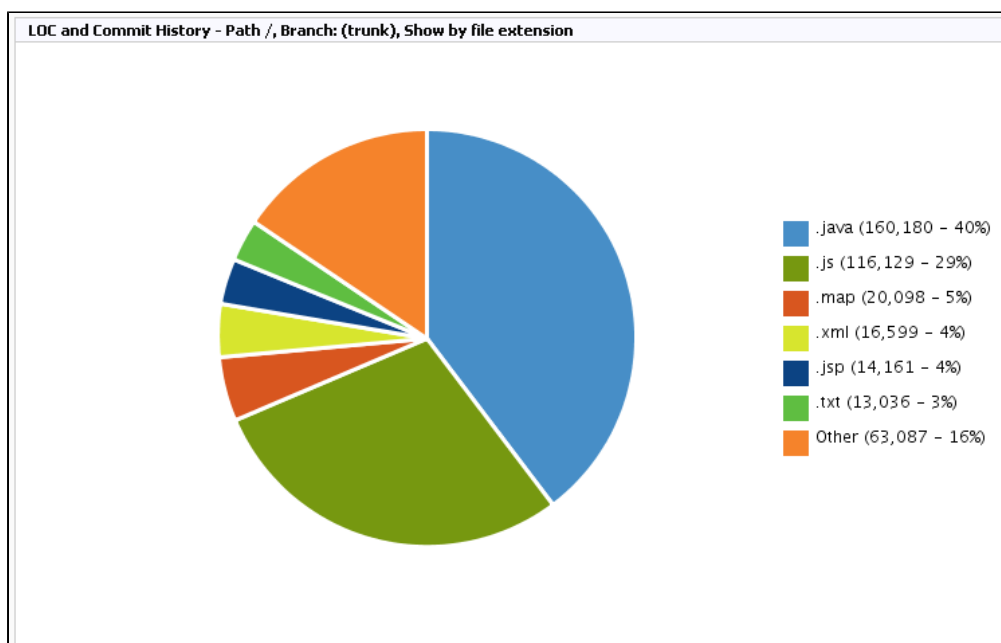
Screenshot: FishEye per-author LOC chart



Screenshot: FishEye per-author LOC chart showing multiple authors



Screenshot: FishEye LOC chart by file extension



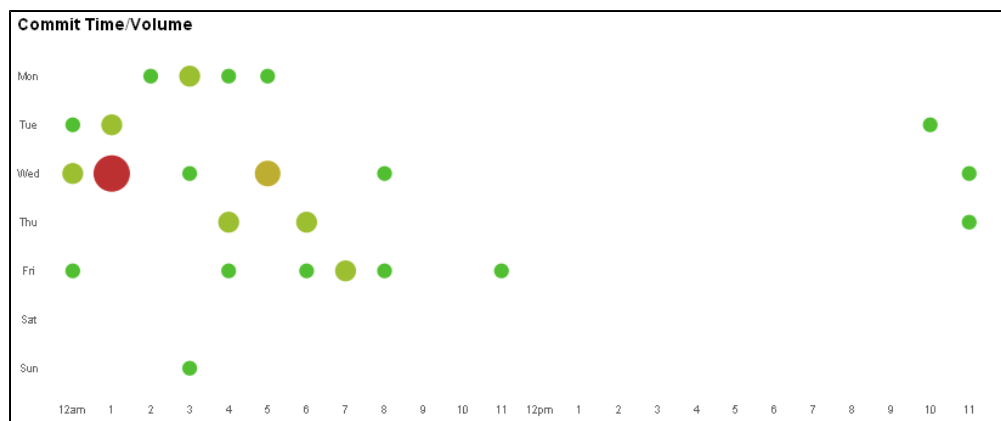
Per-Author Lines of Code Statistics

You can view per-author statistics for lines of code as a chart. This allows you to see how many lines of code were contributed to your project by each author, over time. You can easily view this information on the charts page. Note, if you are upgrading from a previous version of FishEye, you will need to re-index the repository in order to show the per-author information.

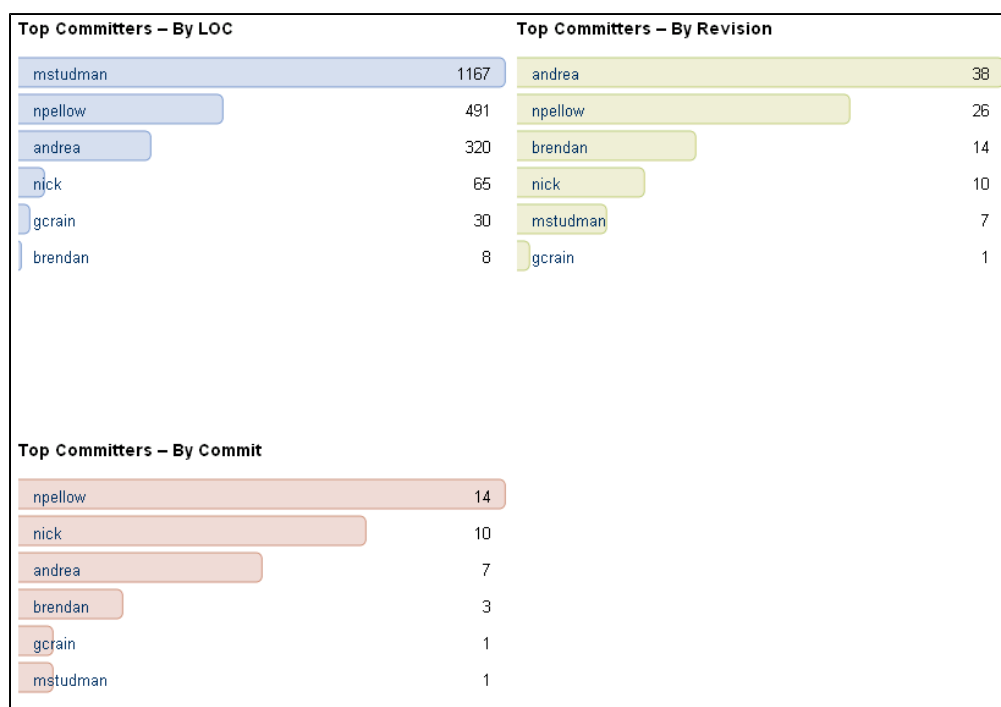
Code Metrics

A number of built-in reports are also provided:

Screenshot: Commit Time/Volume



Screenshot: Top Committers



Notes

Related Topics

[Browsing through a repository](#)

Using favourites in FishEye

FishEye allows you to add changesets, files, people and repositories as favorites. You can view your favorites, or see a stream of all activity relating to your favorites. We suggest that you select items that you are currently working on as favorites, to create a more relevant personalized view.

You can always view your favorites from the menu at the top of the screen, next to your username.

If you are using Crucible, you can also add [code reviews](#) to your favorites.

On this page:

- [Adding favorites](#)
- [Managing favorites](#)

Adding favorites

To add an item to your favorites, follow one of the options below:

People	Hover the mouse cursor over their avatar or username. In the context menu, click Follow .
Changesets	Open the changeset and click the grey star icon next to its name, near the top of the screen.
Files or folders	Open the file or folder and click the grey star icon that appears next to its name. The name appears in the breadcrumb links at the top of the screen.
Repository	Click the Source tab and then the grey star icon that appears next to the name of the desired repository.

Managing favorites

To change the display name for, or remove, a favorite:

1. Choose **Favorites** from your user menu at the top right.
2. Click the yellow star beside a favorite to edit the display name or remove the favorite.

Due to [FE-2348](#) you cannot currently rename favorite directories, users or committers

Changeset discussions

Please see the [Crucible documentation](#) for instructions on this feature.

Viewing the commit graph for a repository

The commit graph shows changesets in their respective branches, using configurable "swimlanes". This allows you to see key information such as branching and merging (and if you are using Git or Mercurial, you will be able to see anonymous branches as well).

The Highlight feature of the commit graph allows you to highlight different types of information in the swimlanes or changeset list:

- ancestors and descendants for a changeset
- commits with JIRA issues
- reviewed and unreviewed changesets.

For example, if you have the **JIRA issues** highlight active, clicking a changeset with a JIRA issue in the commit comment will show all other changesets with the same JIRA issue.

Before you begin:

- Subversion repositories currently do not show lines between branch swimlanes (i.e. merging). But in some cases, FishEye might pick up associations based on SVN branch points.
- Some features of the commit graph are only available if you are using [Crucible](#) with FishEye. For details, see the description below.
- Some features of the commit graph are only available if you are using [JIRA](#) with FishEye. For details, see the description below.

On this page:

- [Viewing the commit graph for a repository](#)
- [Highlighting the lineage of a changeset](#)
- [Highlighting JIRA issues](#)
- [Highlighting reviewed changesets](#)
- [Highlighting commits by an author](#)
- [Highlighting search results](#)
- [Viewing changesets across all branches](#)
- [Reordering swimlanes for Git repositories](#)

Related pages:

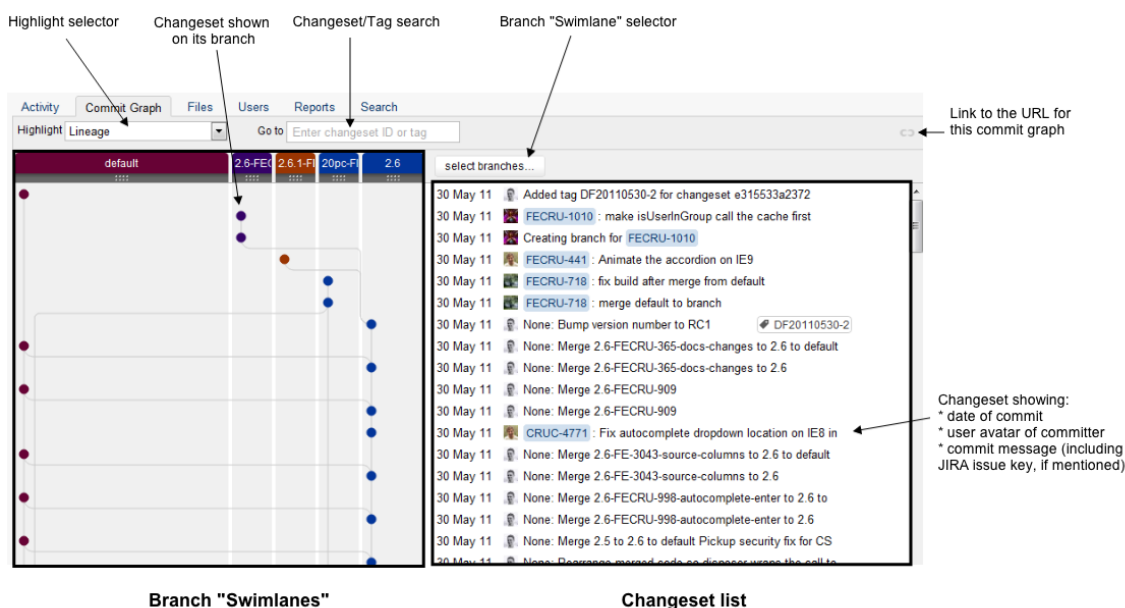
- [Subversion Changeset Parents and Branches](#)
- [What are Subversion root and tag branches?](#)
- [Perforce Changesets and Branches](#)
- [Using the FishEye screens](#)
- [Browsing through a repository](#)
- [JIRA Integration in FishEye](#)

Viewing the commit graph for a repository**To view the commit graph for a repository:**

1. Navigate to the desired repository, as described on [Browsing through a repository](#).
2. Click the **Commit Graph** tab.

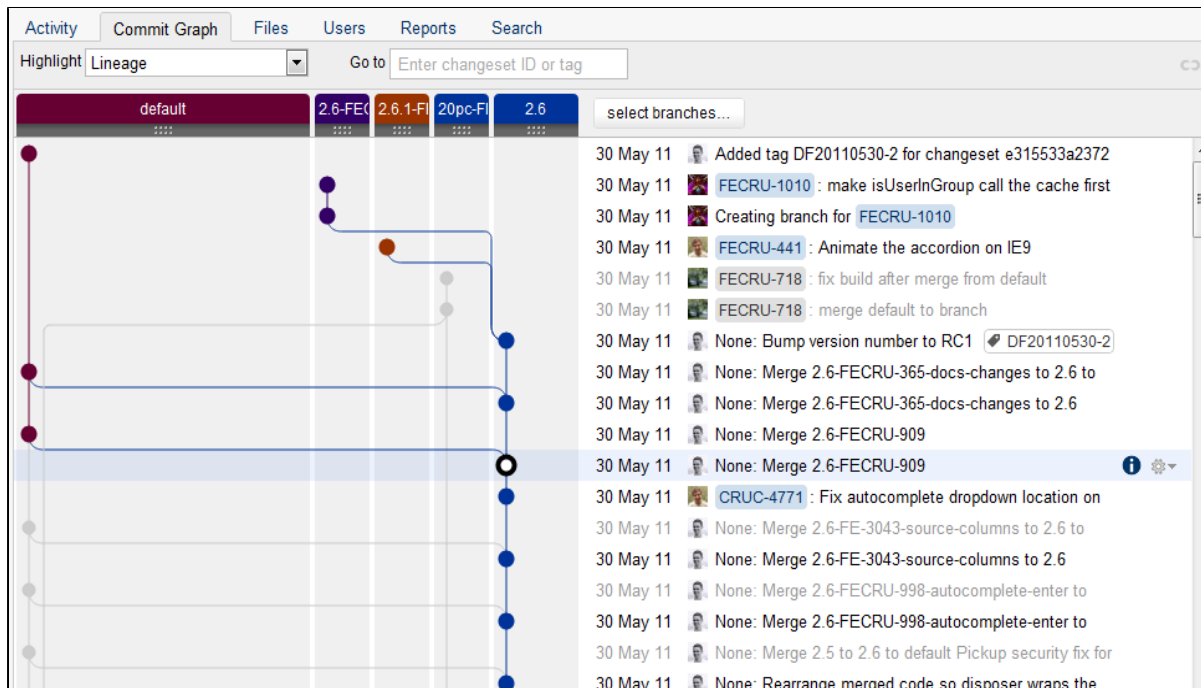
Selecting, or hovering on, a changeset (regardless of highlight) will display the following in the row for the changeset:

- an  icon. Click this icon to see details for the changeset.
- a cog icon with a menu that allows you to see the changeset ID, view the full changeset, view the changeset in the activity stream, or to create a review for the changeset.

**Highlighting the lineage of a changeset**

Choose **Highlight > Lineage** to show the ancestor and descendant changesets for a selected changeset.

Action	Behavior
Click on a changeset in the changeset list	Highlights where a changeset comes from and where it propagates to, i.e. its ancestors and descendants.
Hover over a changeset in a swimlane	Displays the changeset number and all the branches that the changeset is referenced in. This will include branches that you may not have swimlanes displayed for.

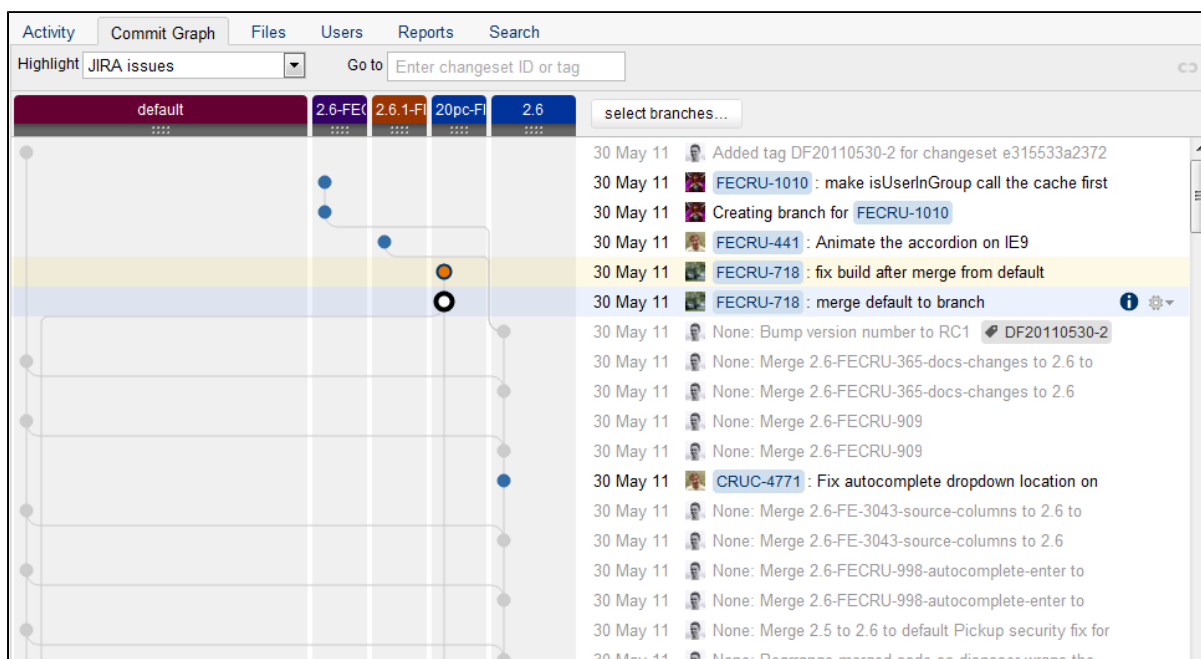


Highlighting JIRA issues

Choose **Highlight** > **JIRA issues** to highlight all the changesets that have a JIRA issue key in the commit message.

This highlight type is only available if you have [integrated FishEye with JIRA](#) and linked your repository to a JIRA project.

Action	Behavior
Click on a changeset in the changeset list	Highlights all other changesets that have the same JIRA issue key in the commit message.
Hover over a changeset in a swimlane	Displays all branches that the changeset is referenced in, and all referenced JIRA issues.



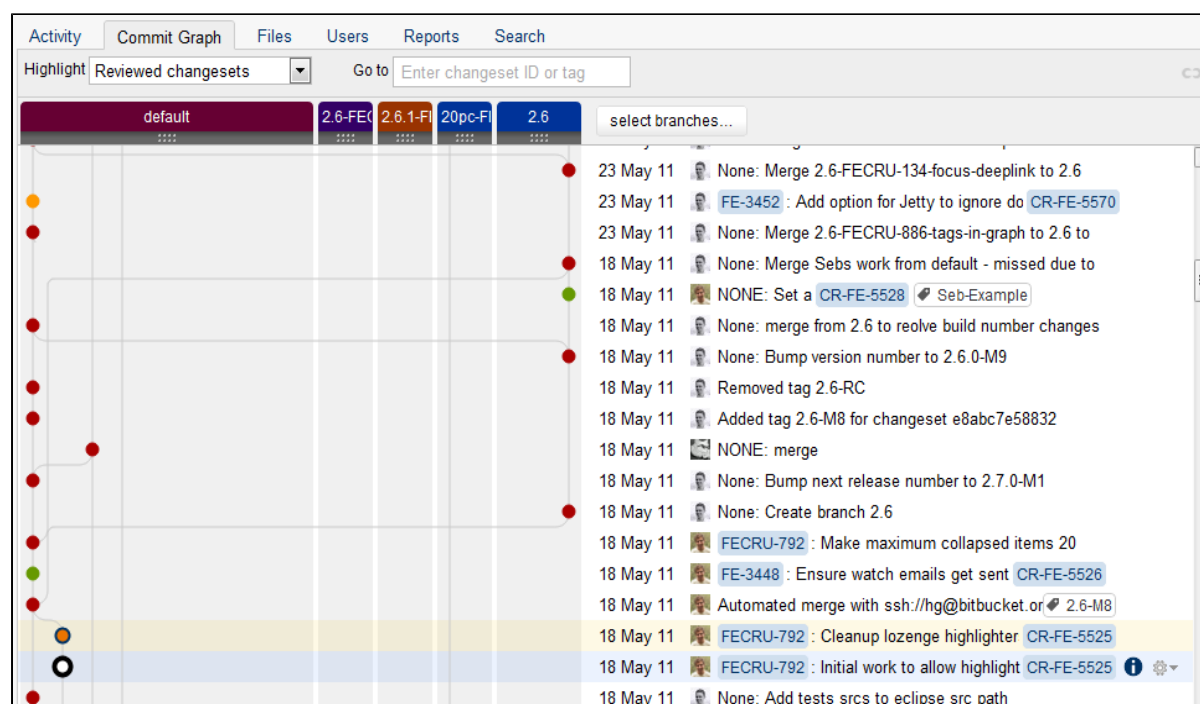
Highlighting reviewed changesets

Choose **Highlight** > **Reviewed changesets** to highlight the changesets that have been reviewed (i.e. included in a Crucible review):

- **Red**: unreviewed, i.e. the changeset is associated with a review in the 'Dead' or 'Rejected' state, or no review is associated.
- **Yellow**: under review, i.e. the changeset is associated with a review not in the 'Dead', 'Rejected' or 'Closed' state.
- **Green**: reviewed, i.e. the changeset is associated with a review in 'Closed' state.

This highlight type is only available if you are using FishEye with Crucible.

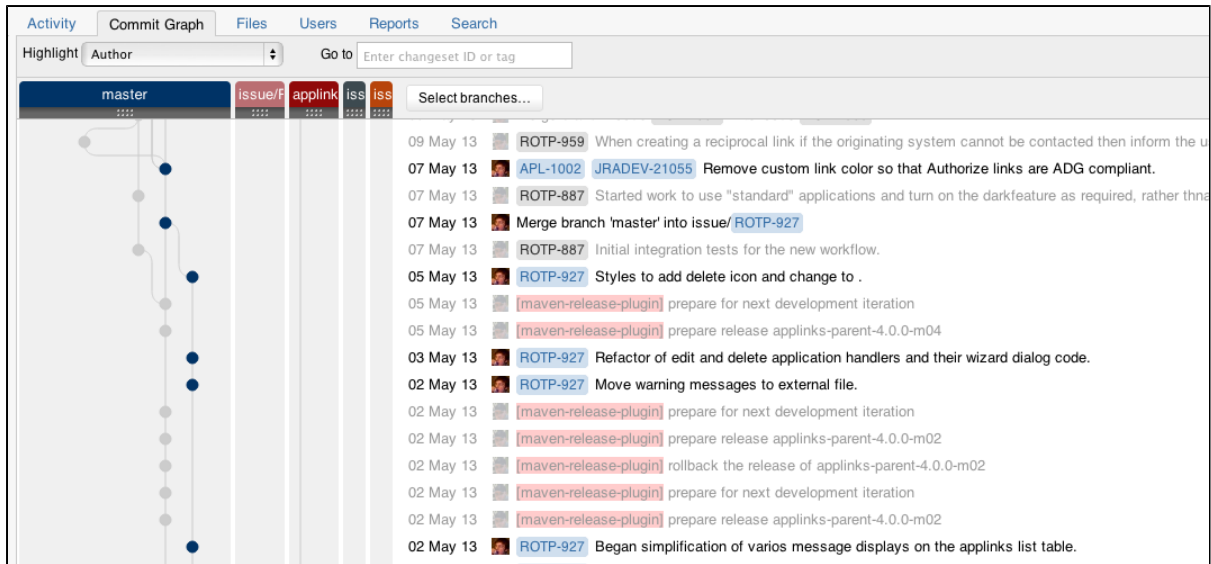
Action	Behavior
Click on a changeset in the changeset list	Highlights the changesets that are part of the same review as the selected changeset.
Hover over a changeset in a swimlane	Displays all branches that the changeset is referenced in, and the Crucible review key.



Highlighting commits by an author

Choose **Highlight** > **Author** to highlight all the changesets submitted by a particular author.

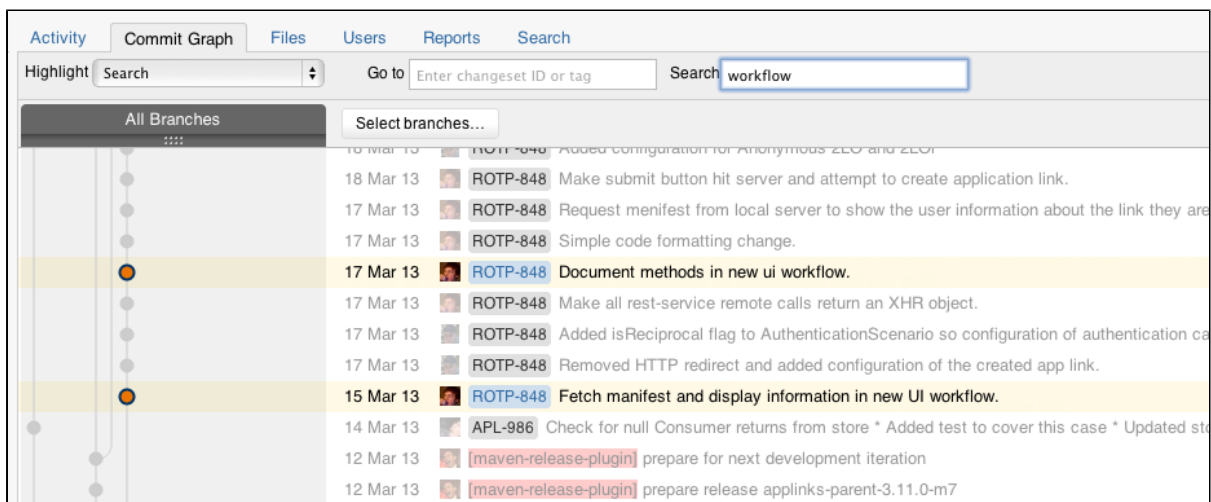
Action	Behavior
Click on a changeset in the changeset list	Highlights the changesets that were submitted by the same author.
Hover over a changeset in a swimlane	Displays the changeset number and all the branches that the changeset is referenced in.



Highlighting search results

Choose **Highlight > Search** to highlight all the changesets where the commit message contains the search term.

Action	Behavior
Click on a changeset in the changeset list	Highlights the changesets that match the search term.
Hover over a changeset in a swimlane	Displays the changeset number and all the branches that the changeset is referenced in.

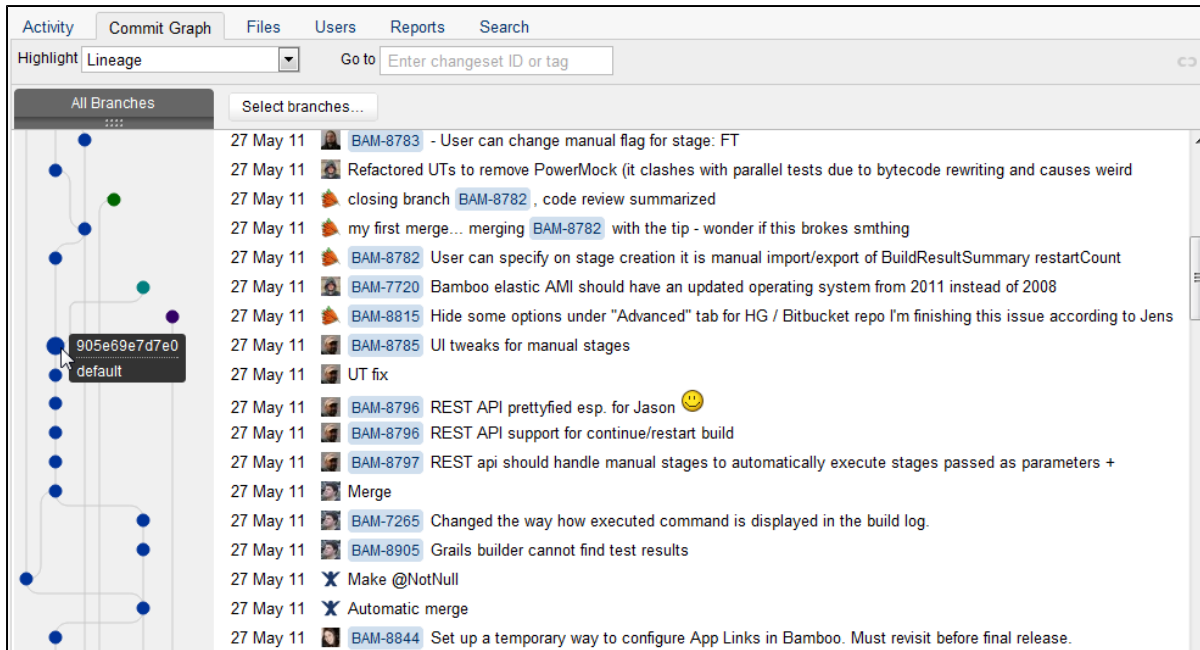


Viewing changesets across all branches

The 'All Branches' mode allows you to view commit activity across all branches of a repository. In this mode, the swimlane headers are not displayed. However, you can hover over any changeset to display information about the changeset, as described in the 'Highlighting Information in the Commit Graph' section above.

To see all the repository's branches in the commit graph:

1. Click **Select branches...** when viewing the commit graph.
2. In the 'Select Branches' dialog, click **Switch to all branches mode**.



Reordering swimlanes for Git repositories

Reordering swimlanes is useful if you just want to show branches in a certain order. However, ordering swimlanes is vital for Git repositories, as it is the only way to determine which branch a commit is from.

When you view the commit graph for a Git repository, FishEye works from the leftmost swimlane to the right and, for each swimlane, checks if the commit is in that branch:

- If the commit is in the branch, a dot is shown representing the commit.
- If the commit is not in the branch, the dot for the commit is moved to the next column on the right.

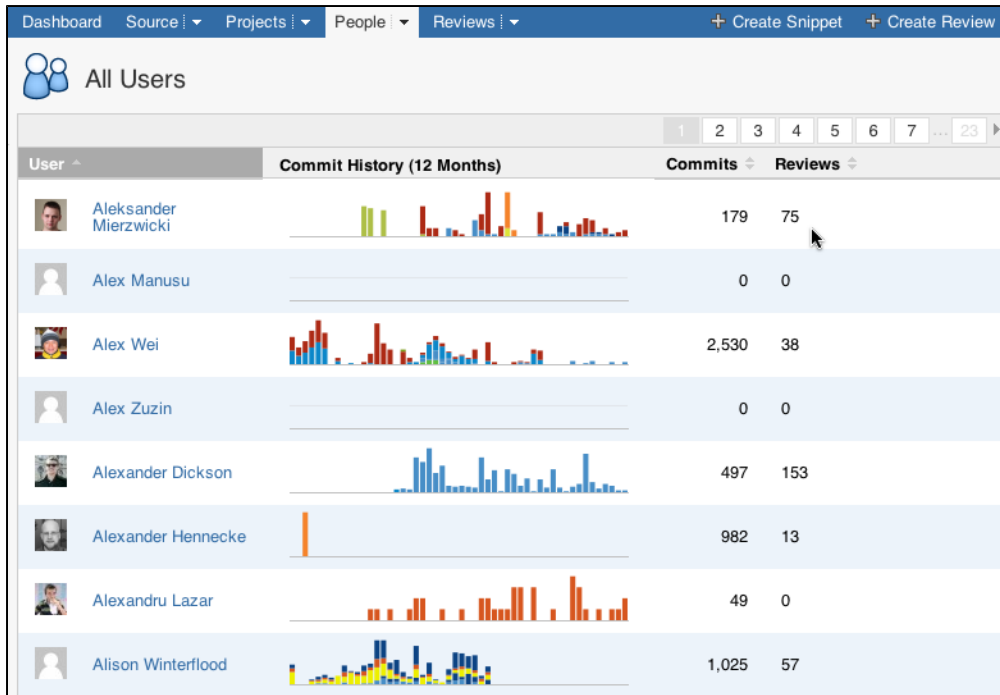
For example, if the 'master' swimlane is to the left of another swimlane, e.g. 'fisheye-2.6' branch, there will be no changesets shown in the 'fisheye-2.6' swimlane, as all the commits will be picked up in the 'master' swimlane. However, if you move the 'fisheye-2.6' swimlane to the left of the 'master' swimlane, it will pick up all of the FishEye 2.6 commits.

For more information, read this Knowledge Base article: [Ordering of Branches Important When Visualizing Git Changeset](#)

Viewing people's statistics

To see charts and activity of everyone who commits code to your FishEye repositories, click the **People** tab at the top of the screen.

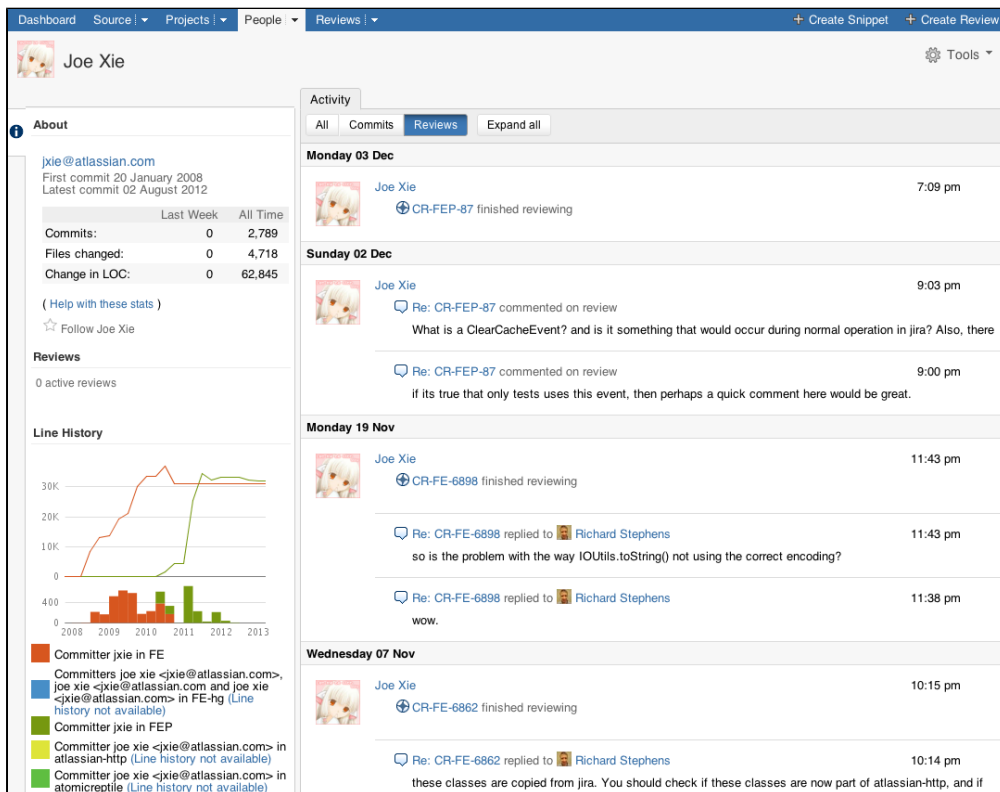
Screenshot: List of all committers in FishEye



The All Users screen shows all those with accounts on the system. You can see their commit history (expressed as a bar graph) and their total number of commits.

Click on a person's name to see detailed information about their additions to the repository, and issue updates. If you are using FishEye with Crucible and have JIRA integration set up, you can see their review activity.

Screenshot: Statistics on a Person in FishEye



Some users may not appear to have the correct number of Files Changed, despite regularly committing. In this situation, if they have committed to a directory which is not covered by the regexes in your

symbolic definition (i.e. they have committed to a directory that is neither trunk, branches or tags) then that directory will be counted as part of trunk. Also note that creating tags and branches themselves does not count toward the totals in FishEye.

Avatars

By default, each user has a unique avatar that is randomly formed from the text in their email address. You can add your own avatar by uploading an image to an external service such as Gravatar, which Crucible supports. See [changing your user profile](#).

If you are using [Crucible](#), statistics for each person's code reviews are also available.

Using Smart Commits

Smart Commits allow repository committers to perform actions such as transitioning JIRA Software issues or creating Crucible code reviews by embedding specific commands into their commit messages.

You can:

- comment on issues
- record time tracking information against issues
- transition issues to any status defined in the JIRA Software project's workflow.

There are other actions available if you use Crucible. See [below](#) for more information.

Each Smart Commit message must not span more than one line (i.e. you cannot use a carriage return in the command), but you can add multiple commands to the same line. See [this example](#) below.

On this page:

- [Smart Commit commands](#)
- [Smart Commit commands for Crucible reviews](#)
- [Advanced examples](#)
- [Get Smart Commits working](#)

Related pages:

- [Enabling Smart Commits](#)
- [Transitioning JIRA issues](#)
- [Write your own Smart Commit](#)

Smart Commit commands

The basic command line syntax for a Smart Commit message is:

```
<ignored text> <ISSUE_KEY> <ignored text> #<COMMAND> <optional  
COMMAND_ARGUMENTS>
```

Any text between the issue key and the Smart Commit command is ignored.

There are three Smart Commit commands you can use in your commit messages:

- [comment](#)
- [time](#)
- [transition](#)

Comment

Description	Adds a comment to a JIRA Software issue.
Syntax	<ignored text> ISSUE_KEY <ignored text> #comment <comment_string>
Example	JRA-34 #comment corrected indent issue
Notes	The committer's email address must match the email address of a single JIRA Software user with permission to comment on issues in that particular project.

Time

Description	Records time tracking information against an issue.
Syntax	<ignored text> ISSUE_KEY <ignored text> #time <value>w <value>d <value>h <value>m <comment_string>
Example	JRA-34 #time 1w 2d 4h 30m Total work logged
Notes	This example records 1 week, 2 days, 4 hours and 30 minutes against the issue, and adds the comment 'Total work logged' in the Work Log tab of the issue. - Each value for w, d, h and m can be a decimal number. - The committer's email address must match the email address of a single JIRA Software user with permission to log work on an issue. - Your system administrator must have enabled time tracking on your JIRA Software instance.

Workflow transitions

Description	Transitions a JIRA Software issue to a particular workflow state.
Syntax	<ignored text> ISSUE_KEY <ignored text> #<transition_name> <comment_string>
Example	JRA-090 #close Fixed this today

Notes	This example executes the close issue workflow transition for the issue and adds the comment 'Fixed this today' to the issue. Note that the comment is added automatically without needing to use the <code>#comment</code> command. You can see the custom commands available for use with Smart Commits by visiting the JIRA Software issue and seeing its available workflow transitions: 1. Open an issue in the project. 2. Click View Workflow (near the issue's Status). The Smart Commit only considers the part of a transition name before the first space. So, for a transition name such as <code>finish work</code>, then specifying <code>#finish</code> is sufficient. You must use hyphens to replace spaces when ambiguity can arise over transition names, for example: <code>#finish-work</code>. If a workflow has two valid transitions, such as: • Start Progress • Start Review A Smart Commit with the action <code>#start</code> is ambiguous because it could mean either of the two transitions. To specify one of these two transitions, fully qualify the transition you want by using either <code>#start-review</code> or <code>#start-progress</code>. • When you resolve an issue with the <code>#resolve</code> command, you cannot set the Resolution field with Smart Commits. • If you want to add a comment during the transition, the transition must have a screen associated with it. • The committer's email address must match the email address of a single JIRA Software user with the appropriate project permissions to transition issues.

Smart Commit commands for Crucible reviews

When creating a new review using a Smart Commit the [default project objectives](#) are added to the review, and since Fisheye/Crucible 2.10.2, the commit message is also copied to the review objectives.

Note that you cannot add arbitrary objectives to the review using a Smart Commit.

Create a review

Description	Create a review in Crucible.
Syntax	<code><commit message> +review <project key></code>
Example	<code>Fix a bug +review CR-TEST</code>
Notes	The <code>+review</code> command creates a new review in the project CR-TEST with the content of the changeset. The review will be in a draft state unless the project has default reviewers or reviewers are explicitly mentioned. If you only have one project in Crucible, or a repository is a project's default repository, it is not necessary to mention the project key.

Add reviewers

Description	Add reviewers when creating a new review in Crucible.
Syntax	<code><commit message> +review <project key> <reviewer_1> <reviewer_2>... <reviewer_n></code>
Example	<code>Fix a bug +review CR-TEST @jcage @skhan</code>

Notes	The command will create a new review in <i>PROJ</i> and add the users <i>jcage</i> and <i>skhan</i> to the review. The review will be automatically started if reviewers are specified. Note that you cannot add reviewers to existing reviews using Smart Commits.

Update a review

Description	Iteratively add new code changes to a review
Syntax	<commit message> +review <review key>
Example	Implement rework on past work +review CR-TEST-123
Notes	Often, reviews require rework and changes in response to comments left by the team. When committing these changes, adding the review key will iteratively add these new changes to the review. Note that: - Each commit command in the Smart Commit must not span more than one line (i.e. you cannot use carriage returns). However, you can use multiple commands in the same commit message, and these can be on the same line. - Creating a review in Crucible using a Smart Commit requires that the author of the changeset has already been mapped to a Crucible username. See 'Author mapping' on Changing your user profile.

Advanced examples

Multiple commands on a single issue

Syntax	<ISSUE_KEY> #<COMMAND_1> <optional COMMAND_1_ARGUMENTS> #<COMMAND_2> <optional COMMAND_2_ARGUMENTS> ... #<COMMAND_n> <optional COMMAND_n_ARGUMENTS>
Commit message	JRA-123 #time 2d 5h #comment Task completed ahead of schedule #resolve
Result	Logs 2 days and 5 hours of work against issue JRA-123, adds the comment 'Task completed ahead of schedule', and resolves the issue.

Multiple commands over multiple lines on a single issue

Syntax	<ISSUE_KEY> #<COMMAND_1> <optional COMMAND_1_ARGUMENTS> #<COMMAND_2> <optional COMMAND_2_ARGUMENTS> ... #<COMMAND_n> <optional COMMAND_n_ARGUMENTS>
Commit message	JRA-123 #comment Imagine that this is a really, and I mean really, long comment #time 2d 5h

Result	Adds the comment 'Imagine that this is a really, and I', but drops the rest of the comment. The work time of 2 days and 5 hours is <i>not</i> logged against the issue because there is no issue key for the #time command in the second line. That is, each line in the commit message must conform to the Smart Commit syntax. This example would work as expected if set out as: <pre>JRA-123 #comment Imagine that this is a really, and I mean really, long comment JRA-123 #time 2d 5h</pre>

A single command on multiple issues

Syntax	<ISSUE_KEY1> <ISSUE_KEY2> <ISSUE_KEY3> #<COMMAND> <optional COMMAND_ARGUMENTS> etc
Commit message	JRA-123 JRA-234 JRA-345 #resolve
Result	Resolves issues JRA-123, JRA-234 and JRA-345. Multiple issue keys must be separated by whitespace or commas.

Multiple commands on multiple issues

Syntax	<ISSUE_KEY1> <ISSUE_KEY2> ... <ISSUE_KEYn> #<COMMAND_1> <optional COMMAND_1_ARGUMENTS> #<COMMAND_2> <optional COMMAND_2_ARGUMENTS> ... #<COMMAND_n> <optional COMMAND_n_ARGUMENTS>
Commit message	JRA-123 JRA-234 JRA-345 #resolve #time 2d 5h #comment Task completed ahead of schedule
Result	Logs 2 days and 5 hours of work against issues JRA-123, JRA-234 and JRA-345, adds the comment 'Task completed ahead of schedule' to all three issues, and resolves all three issues. Multiple issue keys must be separated by whitespace or commas.

Get Smart Commits working

Your FishEye administrator must have:

- Enabled Smart Commits in FishEye. See [Enabling Smart Commits](#).
- Configured an application link between FishEye/Crucible and JIRA Software. See [Linking to another application](#).

Note that:

- Smart Commits only support the default JIRA Software issue key format (that is, two or more uppercase letters, followed by a hyphen and the issue number, for example BAM-123).
- Smart Commits don't provide for field-level updates in JIRA Software issues.
- When using Smart Commits you can use linkers that create a hyperlink to the JIRA Software issue. See [Linkers](#) for more information.
- If there are any errors during the processing of Smart Commits, they will be logged to FishEye's error console, as well as emailed to the actioning users. Speak to your FishEye administrator about [Configuring SMTP](#).

If the application link is configured as OAuth

If the application link to JIRA Software is configured to use OAuth, the committing user must authenticate with JIRA Software before any Smart Commit will work with JIRA Software.

▼ [Click to see how to authenticate with JIRA Software...](#)

1. Create a test review in FishEye.
2. Log in as the committing user.
3. Open the review and click **Edit Details**.
4. Enter a JIRA Software issue key for 'Linked Issue' and click **Link**. You'll be prompted to authenticate.
5. Do this for every committing user (there's no need to create a new review, just link the review to any JIRA Software issue).

Changing your user profile

You can change FishEye (and Crucible) settings such as password, notifications, profile image and display settings.

To change your FishEye settings:

1. Log into FishEye.
2. Choose **Profile settings** from the User Menu (with your avatar) at the top of the screen.
3. Update your user settings as required. Each tab is described in more detail below.
4. Click **Close**.

On this page:

- [Display settings](#)
- [Profile and email](#)
- [Change password](#)
- [OAuth authentication](#)
- [Author mapping](#)
- [Watches](#)
- [Reviews](#)

Display settings

Display Settings	File history view mode	Default is Logical . In Subversion repositories, FishEye is able to show all operations on a single logical file spread across a number of physical paths - i.e. operations in different branches. When this is set to Logical , FishEye will show all the operations across all branches . In Physical mode, only the operations related to the physical path whose history is being viewed are shown.
	Timezone	Default is the timezone of the FishEye server.
Changelog	Changesets per page	The default is 30 per page.
	Always expand changesets in stream	Default is Yes .
Diff view	Diff mode	Default is Unified .
	Line wrapping	Default is None i.e. long lines will never word-wrap. Soft is when long lines will word-wrap.
	Context lines	Default is 3. The number of lines to show (for context), if the diff contains more than three lines of code.

Source view	Tab width	Default is 8. Can be changed to a number between 1 and 10.

Profile and email

Email settings	Display Name	Name displayed for the user currently logged in.
	Email Address	The address to which all email notifications will be sent.
	Email Format	Default is Text . Can be sent as HTML .
Email watches	Send Watch Emails	The frequency at which emails will be sent for watch notifications. Immediately is the default value. Daily sends a summary of changes.
Profile Picture	Choose picture	Upload an avatar image of your choice. This image will be displayed next to your username throughout FishEye/Crucible. Accepted formats are JPG, GIF and PNG. Image file size limit is 2Mb. Images will be automatically cropped on upload. This is disabled if avatars are served from an external server – see Configuring avatar settings.

Change password

Change your password from this tab, if required. Please note that passwords are case-sensitive.

 This tab is not displayed if your FishEye instance is connected to an external LDAP authentication source, such as LDAP. You will need to contact your administrator for assistance.

OAuth authentication

Configure your OAuth settings on this page. You can choose to allow gadgets/applications to access FishEye data using your account.

Read more about [Linking to another application](#).

Author mapping

The **Author Mapping** tab allows you to make an association between you (as a logged-in user) and a committer, for each repository.

This is only necessary if the name or email of the user within FishEye is different from the committer name or email within the repository. By default, FishEye will automatically match users to committers where it can.

Note that FishEye/Crucible administrators can control whether author mapping is available. See [Configuring user managed mappings](#).

Watches

By adding a 'watch', you can ask to receive emails about changes made to the repository. Any watches that you

have set up in FishEye/Crucible will be displayed on this tab. You can watch the dashboard activity stream, [change logs](#) and [repositories](#). Watching an activity stream/repository allows you to receive emails when updates occur. Note, the option to add a watch may only be available if the administrator has [enabled watches](#) for the repository.

You can delete any of your watches by clicking **Delete** next to the watch.

Reviews

This functionality is used by [Crucible](#).

If the SMTP server is set up, then you will receive emails when different actions occur within Crucible.

You can change the options described below, to specify the stages at which emails will be sent.

Auto-mark files as 'read'	Default is Yes .	
Review Notifications Events	State change	Default is Immediate . A Crucible review moves through different states e.g: 'Draft', 'Under Review'. An email is sent when the state changes.
	Comment added	Default is Immediate . An email is sent when a comment is added to a review.
	Comment reply added	Default is Immediate . An email is sent (to the Moderator only) when any reviewer has completed their review .
	Participant finished	Default is Immediate . An email is sent when a reviewer is added or removed from a review, after it has gone into the 'Under Review' state .
	General message	Default is Immediate . An email is sent when a reviewer is added or removed from a review, after it has gone into the 'Under Review' state .
	File revision added	Default is Immediate .
	Uncomplete review if defect is raised:	Default is Yes . This allows reviews to be resurrected automatically to deal with new code or defects.
	Uncomplete review if revision is added:	Default is Yes . This allows reviews to be resurrected automatically to deal with new code or defects.
	My actions	Default is No . If set to Yes , an email is sent every time you perform an action on a review.

Batch Notifications will be sent out by Crucible every 30 minutes. All notifications will be rolled up into a single digest e-mail.

Re-setting your password

If you need to reset your password, FishEye has a mechanism to generate a new password and send it to the email address in your profile.

To reset your password:

1. On the log in screen, click **Unable to access your account?**.
2. Enter your username or email address and complete the [Captcha](#) step. An email will be sent to the email address specified in your profile.
3. Click the link in the email.

On the resulting web page, you will see the message 'A new password has been sent to your account.' An email will arrive in your inbox, containing your new password.

i If you receive a password-reset email that you did not request, simply disregard it to continue using your current password.

Pattern matching guide

FishEye supports a powerful type of regular expression for matching files and directories (same as the pattern matching in Apache Ant).

These expressions use the following wild cards:

?	Matches one character (any character except path separators)
*	Matches zero or more characters (not including path separators)
**	Matches zero or more <i>path segments</i> .

Remember that Ant globs match *paths*, not just simple filenames.

- If the pattern does not start with a path separator i.e. / or \, then the pattern is considered to start with / ** /.
- If the pattern ends with / then ** is automatically appended.

- A pattern can contain any number of wild cards.

Also see the [Ant documentation](#).

Examples

*.txt	Matches /foo.txt and /bar/foo.txt but not /foo.txtty or /bar/foo.txtty/
/*.txt	Matches /foo.txt but not /bar/foo.txt
dir1/file.txt	Matches /dir1/file.txt, /dir3/dir1/file.txt and /dir3/dir2/dir1/file.txt
**/dir1/file.txt	Same as above.
/**/dir1/file.txt	Same as above.
/dir3/**/dir1/file.txt	Matches /dir3/dir1/file.txt and /dir3/dir2/dir1/file.txt but not /dir3/file.txt, /dir1/file.txt
/dir1/**	Matches all files under /dir1/
/dir1*	Matches all files as /dir11, /dir12, /dir12345 and so on.
/dir??	Matches all files as /dir11, /dir22 and so on, replacing just 2 characters.

Date expressions reference guide

FishEye supports a wide variety of date expressions. A date has the two possible general forms:

- DATE[+-]TIMEZONE[+-]DURATION, or
- DATECONSTANT[+-]DURATION.

The TIMEZONE and DURATION parts are both optional.

TIMEZONE can be an offset from GMT HHMM or HH:MM, or simply the letter Z to denote GMT. If no timezone is given, the FishEye server's configured timezone is used.

DATE can be either of the following:

YYYY-MM-DDThh:mm:ss	Specifies a time and date (separated by a T). The seconds part may contain a fractional component. A / can be used instead of - as a separator.
YYYY-MM-DD	Specifies 00:00:00 on the given date. A / can be used instead of - as a separator.

DATECONSTANT can be any of:

now	This very instant (at the time the expression was evaluated).
today todaygmt	The instant at 00:00:00 today. (server-time* or GMT)
thisweek thisweekgmt	The instant at 00:00:00 on the first day of this week. Sunday is considered the first day. (server-time* or GMT)
thismonth thismonthgmt	The instant at 00:00:00 on the first day of this month. (server-time* or GMT)

thisyear thisyeargmt	The instant at 00:00:00 on the first day of this year. (server-time* or GMT)

* The timezone used for server-time is part of the FishEye configuration

The syntax for DURATION is similar to the XML Schema duration type. It has the general form PnYnMnDTnHnMnS. See Section 3.2.6 of the XML Schema Datatypes document for more details.

Examples

2005-01-02	The start of the day on January 1, 2005 (server's timezone)
2005-01-02-0500	The start of the day on January 1, 2005 at GMT offset -0500 (New York)
2005-01-02T12:00:00Z	Midday, January 1, 2005 GMT
today-P1D	Yesterday (start of day)
today+P1D	Start of tomorrow
thismonth-P1M	Start of last month
thisyear+P1Y	Start of next year
now-PT1H	One hour ago
now+PT1H2M3S	One hour, two minutes and three seconds from now

EyeQL reference guide

FishEye contains a powerful query language called **EyeQL**. EyeQL is an intuitive SQL-like language that allows you to write your own specific queries. [See examples](#).

EyeQL allows you to perform complex searches either within the [Advanced Search](#) or incorporated in scripts when programming the FishEye API.

<i>query:</i>	select revisions (from (dir directory) word)? (where clauses)? (order by date (asc desc)?)? Notes: <i>asc</i> produces 'ascending order'. <i>desc</i> produces 'descending order'. (group by (file dir directory csid changeset))? (return return-clauses)? (limit limit-args)?
<i>clauses:</i>	<i>clause ((or and ,) clause)*</i> Notes: <i>and</i> binds more tightly than <i>or</i> . ',' (comma) means 'and'.
<i>clause:</i>	(clauses) not clause path (not)? like word Notes: <i>word</i> is an Antglob.

path = word

Notes:

Defines an exact path without wildcards or variables. **path** must represent a complete (hard-coded) path.

path != word

Notes:

Defines an exact path exclusion without wildcards or variables. **path** must represent a complete (hard-coded) path.

date in ((| [) dateExp, dateExp () |])

Notes: The edges are

inclusive if [or] is used.

exclusive if (or) is used.

date dateop dateExp

Notes:

dateop can be <, >, <=, >=, =, == or !=.

author = word

author in (word-list)

comment matches word

Notes:

Does a full-text search.

comment = string

Notes:

Matches *string* exactly.

Most comments end in a new line, so remember to add \n at the end of your string.

comment =~ string

Notes:

string is a regular expression.

content matches word

Notes:

Does a full-text search.

At this time searches are restricted to HEAD revisions.

(modified|added|deleted)? on branch word

Notes:

Selects all revisions on a branch.

modified excludes the branch-point of a branch.

added selects all revisions on the branch if any revision was added on the branch.

deleted selects all revisions on the branch if any revision was deleted on the branch.

tagged op? word

Notes:

op can be <, >, <=, >=, =, == or !=.

op defaults to == if omitted.

These operators are 'positional' and select revisions that appear on, after, and/or before the given tag.

between tags tag-range

after tag word

before tag word

is head (on word)?

Notes:

This selects the top-most revision on any branch, if no branch is specified.

is (dead | deleted)

Notes:

Means the revision was removed/deleted.

	is added Notes: Means the revision was added (or re-added). csid = word Notes: Selects all revisions for the given changeset ID. p4:jobid = word Notes: finds revisions whose Perforce jobid is <i>word</i>. p4:jobid =~ word Notes: finds revisions whose Perforce jobid matches regex <i>word</i>. reviewed Notes: (<i>applies to Crucible reviews</i>) alias for in or before any closed review. (in before in or before) review word (in before in or before) any (review states)? review Notes: <i>word</i> is a review key. in selects reviewed revisions. If a review contains a diff, then only the most recent revision is considered in the review. before selects all revisions in a file prior to the revision in the review. <i>review states</i> is a comma-separated list of open, closed, draft.
<i>tag-range:</i>	(([] T1:word, T2:word ())) Notes: A range of revisions between those tagged T1 and T2. The edges are: inclusive if [or] is used. exclusive if (or) is used. You can mix edge types. These are all valid: (T1, T2), [T1, T2], (T1, T2] and [T1 , T2). Having trouble with Subversion tags? See How tags work in Subversion for more information.
<i>word:</i>	Any <i>string</i> , or any non-quoted word that does not contain white space or any other separators.
<i>string:</i>	A sequence enclosed in either " (double quotes) or ' (single quotes). The following escapes work: \ ' \ " \ n \ r \ t \ b \ f. Unicode characters can be escaped with \uXXXX. You can also specify strings in 'raw' mode like r "foo". (Similar to Python's raw strings. See Python's own documentation).
<i>dateExp:</i>	See our Date expressions reference guide for more information on date formats.
<i>return-clauses:</i>	return-clause (, return-clause)* A return clause signifies that you want control over what data is returned/displayed.
<i>return-clause:</i>	(path dir directory revision author date comment csid isBinary totalLines linesAdded linesRemoved isAdded isDeleted isCopied isMoved tags reviews aggregate) (as word)? The attribute to return, optionally followed by a name to use for the column. Notes: reviews applies to Crucible reviews.

<i>aggregate-return-field:</i>	(count(revisions) count(<i>binary-field</i>) count(distinct <i>other-field</i>) sum(<i>numeric-field</i>) average(<i>numeric-field</i>) max(<i>numeric-field</i>) min(<i>numeric-field</i>)) The aggregate field to return. Notes: <i>binary-fields</i> are isBinary, isAdded, isDeleted, isCopied, isMoved. e.g. count(isAdded) will return the number of added files. <i>numeric-fields</i> are totalLines, linesAdded, linesRemoved. <i>other-field</i> can be path, dir, author, date, csid, tags or reviews. e.g. count(distinct path) will return the number of unique paths. count(distinct tags) will return the number of unique tags. If a group by is given, give sub-totals for each group. With no group by clause, you can have: • return <i>normal columns</i> • return <i>aggregates</i> With a group by changeset csid clause: • return <i>normal columns</i> • return csid, comment, date, author, <i>aggregates</i> With a group by file path clause: • return <i>normal columns</i> • return path, <i>aggregates</i> With a group by dir directory clause: • return <i>normal columns</i> • return dir, <i>aggregates</i> i.e. The EyeQL can contain a returns clause that contains all non-aggregate columns, or all aggregate columns. Non-aggregate and aggregate columns can only be mixed if the columns are unique for the grouping.
<i>limit-clause:</i>	(<i>length</i> <i>offset</i>, <i>length</i> <i>length</i> offset <i>offset</i>) Notes: Limits the number of results to return. <i>offset</i> specifies the starting point of the truncated result set and <i>length</i> specifies the set length. <i>offset</i> is zero-based.

Examples

The following examples demonstrate using EyeQL to extract information from your repository.

Find files removed on the Ant 1.5 branch:

```
select revisions where modified on branch ANT_15_BRANCH and is dead group by changeset
```

As above, but just return the person and time the files were deleted:

```
select revisions where modified on branch ANT_15_BRANCH and is dead return path, author, date
```

Find files on branch and exclude delete files:

```
select revisions where modified on branch ANT_15_BRANCH and not is deleted group by changeset
```

Find changes made to Ant 1.5.x after 1.5FINAL:

select revisions where on branch ANT_15_BRANCH and after tag ANT_MAIN_15FINAL group by changeset

Find changes made between Ant 1.5 and 1.5.1:

select revisions where between tags (ANT_MAIN_15FINAL, ANT_151_FINAL] group by changeset

As above, but show the history of each file separately:

select revisions where between tags (ANT_MAIN_15FINAL, ANT_151_FINAL] group by file

Find Java files that are tagged ANT_151_FINAL and are head on the ANT_15_BRANCH: (i.e. files that haven't changed in 1.5.x since 1.5.1)

select revisions from dir /src/main where is head and tagged ANT_151_FINAL and on branch ANT_15_BRANCH and path like *.java group by changeset

Find changes made by conor to Ant 1.5.x since 1.5.0:

select revisions where between tags (ANT_MAIN_15FINAL, ANT_154] and author = conor group by changeset

Find commits that do not have comments:

select revisions from dir / where comment = "" group by changeset

Find the 10 most recent revisions:

select revisions order by date desc limit 10

Find the 5th, 6th & 7th revisions:

select revisions order by date limit 4, 3

Find commits between two dates:

select revisions where date in [2008-03-08, 2008-04-08]

Find revisions that do not have any associated review:

select revisions where (not in any review)

Return number of matched revisions, the number of files modified, authors who modified code, changesets, tags, and reviews:

```
select revisions
where date in [ 2003-10-10, 2004-12-12 ]
return count(revisions), count(distinct path), count(distinct author),
count(distinct csid), count(distinct tags), count(distinct reviews)
```

As Sub-totals for each distinct changeset, Return csid, the author, date, comment, number of matched revisions, the number of files modified, the lines added/removed:

```
select revisions
where date in [ 2003-10-10, 2004-12-12 ]
group by changeset
return csid, author, date, comment, count(revisions), count(distinct
path), sum(linesAdded), sum(linesRemoved)
```

For each matched file, return the file name, number of matched revisions, the lines added/removed:

```
select revisions
where date in [ 2003-10-10, 2004-12-12 ]
group by file
return path, count(revisions), sum(linesAdded), sum(linesRemoved)
```

Show all the changesets that are not in or before any closed review:

```
select revisions
from dir /
where not reviewed
group by changeset
return csid, author, count(revisions), comment
```

Includes changesets that were explicitly in a review

```
select revisions
where not in any closed review and not in any open review
group by changeset
return path, revision, author, date, csid, reviews
```

Integrating FishEye with Atlassian applications

You can integrate FishEye with the following Atlassian applications:

	When FishEye is integrated with a JIRA application you can: • Use smart commits to transition JIRA issues • Delegate user and group management to the JIRA application When Crucible is integrated with JIRA, you can: • Transition JIRA application issues When a JIRA application is integrated with FishEye, you can: • View an issue's FishEye changesets • Add the FishEye Charts Gadget to your JIRA application dashboard • Add the FishEye Recent Changesets Gadget to your JIRA application dashboard • View an issue's Crucible reviews • Add the Crucible Charts Gadget See Configuring development tools for detailed information.
	When FishEye is used with Crucible, you can: <i>In FishEye:</i> • Use smart commits to create Crucible reviews, add reviews to new reviews and update an existing review. <i>In Crucible:</i> • When using iterative reviews in Crucible, you will be prompted when a new version of a file is available. • Files and changesets displayed in activity streams (e.g. the dashboard activity stream) render as links to the relevant files/changesets. • See your content roots and repositories associated with projects. • See repository lists and browse repositories using the Files tab • View charts or code metrics. See Crucible and FishEye.

	When FishEye is integrated with Bitbucket Server: - You can easily add Bitbucket Server repositories to FishEye with a single click. Once added, the repository behaves just like a native repository in FishEye, so your team gets all the benefits of FishEye indexing, browsing and searching. Furthermore, the repository becomes available to Crucible, so you can perform in-depth code reviews for changes in the repository. - A push to a Bitbucket Server repository that has been added to FishEye automatically triggers FishEye to run an incremental index. You don't have to configure polling for new commits, or set up dedicated FishEye web hooks in your Bitbucket Server instance.
	See: Integrating Crowd with Atlassian FishEye Integrating Crowd with Atlassian Crucible
	When FishEye is integrated with your Bamboo continuous integration server, you can view the code changes that triggered a build. When a build fails due to a compilation error or failed test, you can explore the failed build in FishEye and jump directly into the changeset that broke the build. You can view the history of that changeset to see what the author was trying to fix, take advantage of the side-by-side diff view to analyze the change and then open the correct files in your IDE. For more details see Viewing the Code Changes that Triggered a Build.

JIRA Integration in FishEye

When FishEye is integrated with JIRA Software, you and your team get all the benefits described on this page. Go straight to [Linking FishEye to JIRA Software](#) if you want to connect FishEye to a JIRA Software server.

You can also use JIRA Software for delegated management of your FishEye users. See [JIRA and Crowd authentication](#).

See [Configuring development tools](#) for the full story of how Atlassian tools work together to give you a fast and guided software development process.

Starting in JIRA 6.2.2 the Source and Reviews tabs are only displayed if JIRA Software is unable to display the associated information in the Development Tools panel.

Related pages:

- [Linking FishEye to JIRA Software](#)
- [Enabling Smart Commits](#)

Your user tiers don't need to match between JIRA Software and FishEye/Crucible in order to integrate them. JIRA Software users that are not FishEye users will see the same view as FishEye users within JIRA Software, but will not be able to log in to FishEye to view the source/reviews.

Check development progress of a version in JIRA Software

FISHEYE 3.3+

JIRA 6.4+

The Release Hub in JIRA Software shows the relevant issues and development information for a version – so you can determine which issues are likely to ship at a glance. With JIRA Software and FishEye connected, the release page can warn you about potential development issues that could cause problems for your release.

Version 2.0 UNRELEASED

Start: 03/Nov/14
Release: 15/Feb/15
[Release Notes](#)

48 Warnings

273 Issues in version

262 Issues done

7 Issues in progress

4 Issues to do

Warnings indicate when the status of a JIRA issue doesn't reflect related development activity. For example: an issue marked complete that has an open pull request should be marked as still being in progress.

Unreviewed Code
These issues have been marked complete but the commits are not part of a pull request or review.

1–4 of 4

P	T	Key	Summary	Assignee
		SSP-1663	UI is not loading in IE8	Andrew Swar
		SSP-1979	Incorrect permissions for new report	Andrew Swar
		SSP-1555	As a developer, I want to view the build status for an issue	Bruce Temple
		SSP-1660	Missing error message when Bamboo is unavailable	Eduardo Soai

1–4 of 4

From the Release Hub you can also:

- Release a version
- Mark a version as complete
- Move incomplete issues to other versions
- Trigger release builds (if JIRA is connected to Bamboo)
- Warnings that help you reconcile what is happening in development with JIRA Software data.

To view the Release Hub (with the project sidebar enabled), navigate to a project, click on **Releases**, then select a version listed. See [Checking the progress of a version](#) for more detailed information about using the Release Hub in JIRA Software.

See in JIRA Software the FishEye repository branches related to an issue

FISHEYE 3.3+

JIRA 6.2+

For FishEye 3.3 and later, the FishEye repository branches related to a JIRA Software issue are summarized in the Development panel for the issue, when JIRA Software and FishEye are connected with an [application link](#). To see details of the branches, simply click the **branches** link. You can see which repository each branch is in and when the last commit was made. As long as the issue key is included in the branch name the branch is automatically linked to the JIRA Software issue.

FUSE-601: 2 branches		
Bitbucket 1 FishEye / Crucible 1		
Repository	Branch	Last commit
FE-hg	3.3-FUSE-601-more-commit-details	10f3418

See in JIRA Software the commits related to an issue

FISHEYE 3.3+

JIRA 6.2+

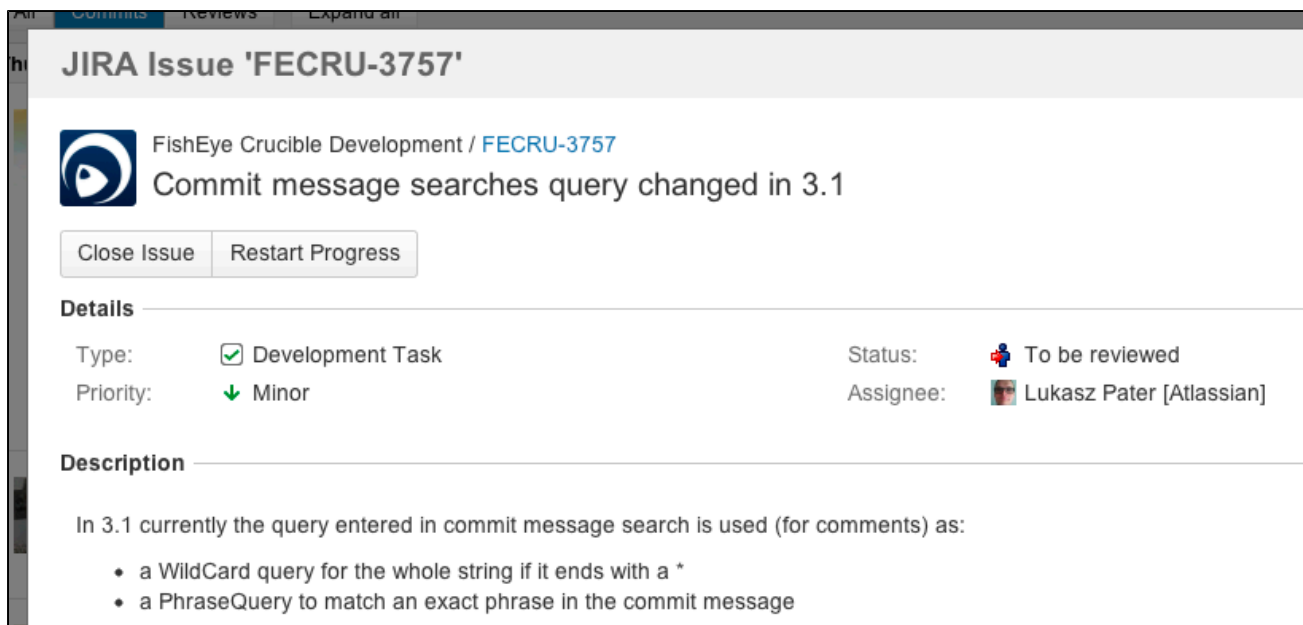
For FishEye 3.3 and later, the FishEye repository commits related to a JIRA Software issue are summarized in the Development panel for the issue, when JIRA Software and FishEye are connected with an [application link](#). You can click the **commits** link to see detailed information such as who made each commit, when they committed, and how many files were changed. Click through to see a particular commit in the FishEye instance where the commit was made. A developer only needs to add the issue key to the commit message for that commit to be automatically linked to the JIRA Software issue.

FUSE-113: 62 unique commits (and 56 duplicates)						
Bitbucket 40 FishEye / Crucible 56 Stash 22						
bamboo-git (atlaseye) Create						
Author	Commit	Message	Reviews	Date	File	
	9b9b93b	FUSE-113 Functional tests	55	05/Dec/13 11:24 AM	8 fil	
	cfba99b	FUSE-113 FT for anonymous access	81	18/Nov/13 1:27 PM	3 fil	
	3dea4f4	FUSE-113 post review	81	18/Nov/13 11:00 AM	1 fil	
	2b69f9e	FUSE-113 post review	2	18/Nov/13 11:00 AM	1 fil	
	563fa3b	FUSE-113 post review	81	18/Nov/13 10:59 AM	1 fil	

See the JIRA Software issues related to commits

FISHEYE 3.1+

FishEye recognizes JIRA Software issue keys, and displays those as links in places such as the activity stream, side-by-side diffs, and commit messages:



JIRA Issue 'FECRU-3757'

FishEye Crucible Development / [FECRU-3757](#)

Commit message searches query changed in 3.1

[Close Issue](#) [Restart Progress](#)

Details

Type: ☒ Development Task Status: To be reviewed
Priority: Minor Assignee: Lukasz Pater [Atlassian]

Description

In 3.1 currently the query entered in commit message search is used (for comments) as:

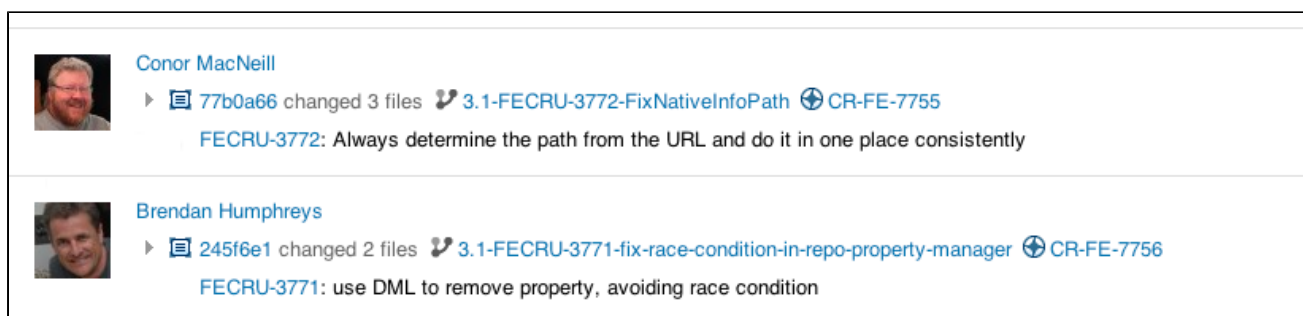
- a WildCard query for the whole string if it ends with a *
- a PhraseQuery to match an exact phrase in the commit message

Click on the linked issue key to see details for the issue, as described next.

See the details for JIRA Software issues

FISHEYE 3.1+

Click a linked issue key anywhere in FishEye to see the details of that issue in a dialog. And you can click the issue key at the top of the dialog to go straight to the issue in JIRA Software:



[Conor MacNeill](#)

▶ [77b0a66](#) changed 3 files [3.1-FECRU-3772-FixNativeInfoPath](#) [CR-FE-7755](#)

[FECRU-3772](#): Always determine the path from the URL and do it in one place consistently

[Brendan Humphreys](#)

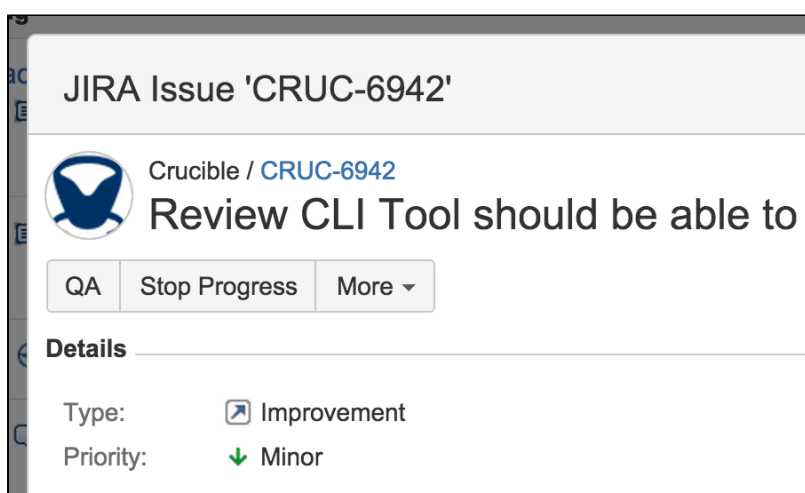
▶ [245f6e1](#) changed 2 files [3.1-FECRU-3771-fix-race-condition-in-repo-property-manager](#) [CR-FE-7756](#)

[FECRU-3771](#): use DML to remove property, avoiding race condition

Transition JIRA Software issues from within FishEye

FISHEYE 3.1+

You can easily [transition](#) a JIRA issue from within FishEye. For example, when viewing a commit, you may want to transition the related JIRA Software issue into QA. Click on a linked JIRA issue anywhere in FishEye to see a dialog with the available workflow steps:



JIRA Issue 'CRUC-6942'

Crucible / [CRUC-6942](#)

Review CLI Tool should be able to

[QA](#) [Stop Progress](#) [More ▾](#)

Details

Type: Improvement
Priority: Minor

Click on a step in the dialog, and complete any displayed fields as required. If there are custom required fields that are unsupported by FishEye, just click **Edit this field in JIRA** to transition the issue directly in JIRA.

See issues from multiple instances of JIRA Software

FISHEYE 3.1+

FishEye can link to more than one JIRA server at a time, so different teams can work with their own projects in different JIRA Software instances, or a single team can link to issues across multiple JIRA Software servers.

Integrating FishEye with Bitbucket Server

This page...

... describes the benefits for you and your team, when FishEye is integrated with [Bitbucket Server](#) (formerly known as Stash).

Set it up...

... with our short guide to help FishEye admins [connect FishEye to Bitbucket Server](#).

Bitbucket Server is...

... Atlassian's on-premise Git repository management solution for enterprise teams. Read about [getting started with Git and Bitbucket Server](#).

Easily keep FishEye repositories synced with Bitbucket Server

FISHEYE 3.5 +

STASH 3.1 +

Once you add a Bitbucket Server Git repository to FishEye, a push to the repository automatically triggers FishEye to run an incremental index. You don't have to configure polling to detect new commits, or set up dedicated FishEye web hooks in your Bitbucket Server instance.

A FishEye administrator only needs to set up an [application link](#) with Bitbucket Server for FishEye to be ready to respond to 'refs changed' notifications published by Bitbucket Server.

Note that FishEye may not receive the notifications published by Bitbucket Server under some circumstances, for example if FishEye is being restarted, or if there are intermittent network issues. Therefore, we don't recommend disabling polling completely, but that the polling period be extended to 2 hours or even 24 hours, so FishEye can catch up with any missed events. See [Updater](#).

Note also that Bitbucket Server was formerly known as Atlassian Stash.

Easily add Bitbucket Server repositories to FishEye

FISHEYE 3.4 +

STASH 2.11.4 +

When FishEye is integrated with Bitbucket Server, a FishEye administrator can easily add Bitbucket Server repositories to FishEye with a single click. Once added, the repository behaves just like a native repository in FishEye, so your team gets all the benefits of FishEye indexing, browsing and searching. Furthermore, the repository becomes available to Crucible (when integrated), so you can perform in-depth code reviews for changes in the repository.

For detailed instructions for adding a Bitbucket Server repository to FishEye, see [Adding an external repository](#).

Transitioning issues in JIRA

Transition a JIRA Software issue from within FishEye in any of the following ways:

- [Transition an issue automatically](#)
- [Transition an issue manually from FishEye](#)
- [Transition an issue with your commit message](#)

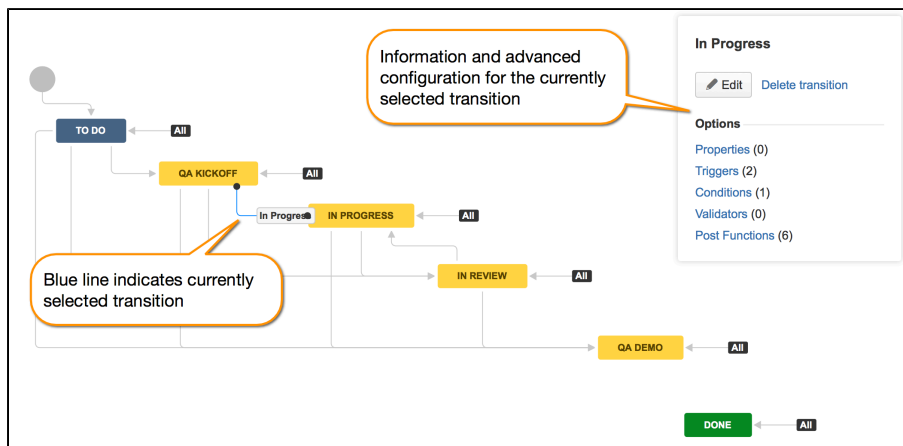
Transition an issue automatically

FISHEYE 3.5.2+

JIRA 6.3.3+

Your JIRA Software workflow can now respond to events in your linked

development tools. For example, when a commit is pushed, your JIRA Software workflow can be configured to automatically transition the related issue. Configure this using triggers for transitions within the JIRA Software workflow editor.



The events published by FishEye are:

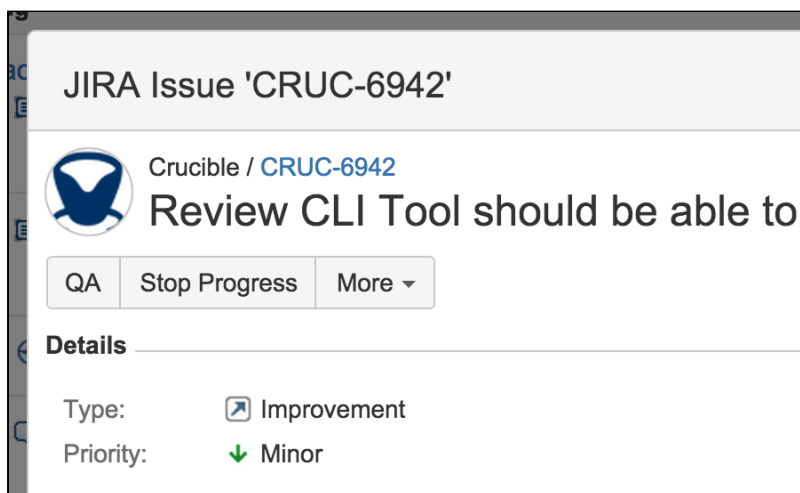
- Commit created
- Branch created

A FishEye administrator can disable event publishing if required. See [JIRA workflow triggers](#).

Transition an issue manually from FishEye

FISHEYE 3.1+

You can easily [transition](#) a JIRA Software issue from within FishEye. For example, when viewing a commit, you may want to transition the related JIRA Software issue to the QA status. Click on a linked JIRA Software issue key in FishEye to see a dialog with the available workflow steps:



Click on a step in the dialog, and complete any displayed fields as required. If there are custom required fields that are unsupported by FishEye, just click **Edit this field in JIRA Software** to go to JIRA Software to transition the issue.

Transition an issue with your commit message

FISHEYE 2.7+

Use Smart Commits when you are committing to a repository hosted in FishEye to transition JIRA Software issues. Smart commits allow you to embed commands into your commit messages, which FishEye detects and actions.

For example, if you include the following text in your commit message, FishEye will add the comment 'Task completed ahead of schedule' and resolve the issue, when you perform your commit:

```
JRA-123 #comment Task completed ahead of schedule  
#resolve
```

Read more about [Using Smart Commits](#).

Administering FishEye

The FishEye Administration area allows you to administer your FishEye instance and manage your repositories, users and back-end settings.

To log in to the Admin area, you can either:

- log in to FishEye using an administrator's account.
- click **Administration** at the foot of the screen and enter the admin password.
- navigate to <http://HOSTNAME:8060/admin/>, where HOSTNAME is the name of the server on which you installed FishEye.

Once logged in as an [administrator](#) you can get to the Admin area by clicking the 'cog' menu in the FishEye/Crucible header, and choosing **Administration**.

You will also want to read about the [command-line options](#) for controlling FishEye.

You can disable FishEye's Administration Console by setting `admin-hash=""` in the `<config>` element of `config.xml` before starting FishEye.

We recommend that you run FishEye as a user that has only **read** access to your repository.

If you're running FishEye on Windows you might want to install it as a service so that FishEye starts up automatically during start-up. See [Run FishEye as a Windows Service](#).

If you intend to use FishEye with [Subversion](#), please ensure you read the [Supported Platforms](#), [Subversion client setup](#), and [granting permission to FishEye](#) to scan your repository.

If you intend to use FishEye with [Perforce](#), please ensure you read about [Perforce client setup](#).

FishEye runs its own HTTP web server, and additionally listens on a socket for administration/shutdown commands. These default to `:8060` and `127.0.0.1:8059` respectively. You can change both of these addresses by editing `config.xml`. You'll need to restart FishEye after doing that.

If you need to reset the administrator password, delete the `admin-hash` attribute in the `<config>` element. You will be prompted to enter an administrator password next time you start FishEye. See [How to reset the Admin password in FishEye or Crucible](#).

One of your first steps will be to [add a repository](#).

Topics

- [Managing your repositories](#)
- [Managing users and groups in FishEye](#)
- [External user directories](#)
- [Migrating to an external database](#)
- [Configuring SMTP](#)
- [Backing up and restoring FishEye data](#)
- [Managing add-ons](#)

- [Linking FishEye to JIRA](#)
- [Linking to another application](#)
- [System settings](#)
- [Configuring FishEye security](#)
- [Tuning FishEye performance](#)
- [How FishEye uses memory](#)
- [Contacting Support](#)

Managing your repositories

FishEye administrators can see all current repositories in the FishEye/Crucible instance in the [Admin area](#). Note that users can see a similar list, without the admin functions, by clicking **Source** in the header.

The list of repositories provides admin functions such as starting and stopping repository scans, disabling and enabling repositories, deleting repositories and configuring various repository settings. You can also add new repositories from the repository list page.

On this page:

- [Viewing repositories](#)
- [Configuring repositories](#)
- [Adding a new repository](#)

Viewing repositories

To view the repositories set up for your FishEye/Crucible instance, log in to the [Admin area](#) and click **Repositories** (under 'Repository Settings').

Use the filter in the table header to search for repositories by attributes in the table columns (Name, Type, Description, Location or State). For example you could filter on all the repositories named "foo" by typing "foo" or you could filter on the disabled repos by typing "disabled". Filtering is not case sensitive.

Use the paging controls at the top right of the table to browse all the available repositories.

Click **Refresh** in the table header to update the list of repositories. See [Updater](#) for information about polling the repositories.

Screenshot: The repository list in the FishEye administration screen

Repositories

Native Repository Access (82)

Plugins (4)

FishEye and Crucible natively supports Subversion, CVS, Perforce, Git and Mercurial repositories ?

Add repository

Refresh

Filter repositories

Page 1 of 5

Name	Type	Description	Location	State	Updates	Actions
 activeobjects	Subversion	a	https://svn.java.net/svn/activeobjects~svn	Running	24 minutes ago	 ▾
 applinks	Subversion		https://studio.atlassian.com/svn/APL	Disabled		 ▾
 applinks-git	Git		git@bitbucket.org:atlassian/application-links.git/	Running	32 seconds ago	 ▾

Configuring repositories

On the Repositories screen, click the name of the repository to see, and edit, the [repository options](#).

Click the cog icon



for a repository to go to the following operations:

- **View** — View the repository options. See [Repository options](#).
- **Browse** (*Running repositories only*) — Browse a repository. See [Browsing through a repository](#).
- **Stop** (*Running repositories only*) — Stop access to the repository and stop the [repository scan](#), if a scan is in progress. See [Operations](#).
- **Start** (*Stopped repositories only*) — Allow access to a stopped repository and start the [repository scan](#). See [Operations](#).
- **Restart** (*Running repositories only*) — Allow access to the repository and resume the [repository scan](#), if necessary. See [Operations](#).
- **Disable** (*Stopped repositories only*) — Disable a previously stopped repository. See [Operations](#).
- **Enable** (*Disabled repositories only*) — Enable the disabled repository. See [Operations](#).
- **Delete** (*Disabled and stopped repositories only*) — Remove the *FishEye profile* for the disabled repository. See [Operations](#).
- **Application Links** — Configure [application links](#) for the repository.

To configure the default options for all repositories, click **Defaults** (under 'Repository Settings') in the left-hand nav panel. See [Repository Options](#) for more information.

Adding a new repository

On the Repositories screen, click **Add repository** in the table header to add a [new repository](#).

Adding an external repository

One of the key actions you perform with FishEye is to add a new external repository.

When you add a repository, and **enable** it, FishEye builds an index and cache, which may take some time to complete.

Once a repository has been added, you can then browse and search it, and further configuration options become available, depending upon the type of repository.

Add a repository

1. Click the 'cog' menu in the FishEye header, and choose **Administration** (you'll need to be logged in as an [administrator](#) to see this link).
2. Click **Repositories** (under 'Repository Settings').
3. Click **Native repository access**, and then **Add repository**.
4. Complete the wizard. See the following topics for more information:
 - [CVS](#)
 - [Git](#)
 - [Mercurial](#)
 - [Perforce](#)
 - [Subversion](#)
5. Click **Add** to finish.

Add a Git repository that is hosted in Bitbucket Server

You can add repositories that are hosted in [Bitbucket Server](#) to FishEye, when these servers are connected using an [application link](#). See [Integrating FishEye with Bitbucket Server](#) for an overview.

When Bitbucket Server is integrated with FishEye:

- You can easily add Git repositories to FishEye. The repository behaves just like a native repository in FishEye, so your team gets all the benefits of FishEye indexing, browsing and searching.
- The repository becomes available to Crucible (when integrated), so you can perform in-depth code reviews for changes in the repository.
- When you add a Bitbucket Server repository to FishEye, a push to the repository will by default automatically trigger FishEye to run an incremental index. No further configuration is required – you don't have to configure polling for new commits, or set up dedicated FishEye web hooks in your Bitbucket Server instance.

You'll need to have an account in the Bitbucket Server instance, as well as permission to view the repository that

you want to add.

1. Click the 'cog' menu in the FishEye header, and choose **Administration** (you'll need to be logged in as an [administrator](#) to see this link).
2. Click **Repositories** (under 'Repository Settings').
3. Click the **Bitbucket repositories** tab, and authenticate with Bitbucket Server if necessary.
4. If multiple instances of Bitbucket Server are [connected](#) to FishEye, use the **Bitbucket server** list to choose the instance of Bitbucket Server that hosts the repository you wish to add.
5. Optionally, type a filter pattern to restrict the list of displayed repositories to those with a matching name, key or project.
6. Click **Add** for each repository that you wish to add to FishEye.

When you add a Bitbucket Server repository:

- FishEye creates a read-only SSH key and adds that as an [access key](#) to the repository in Bitbucket Server. If this operation fails, the key will be added as a user key to your profile in Bitbucket Server. You can check if an access key was added by viewing the repository's settings in Bitbucket Server. See [Using SSH keys to secure Git operations](#) in the Bitbucket Server documentation for more information.

CVS

When [adding](#) or [configuring](#) a CVS (Concurrent Versions System) repository, you can:

- Define repository details, **as described below**.
- Set FishEye's [repository options](#).

There are also the following CVS-specific actions:

- [Updater](#).
- [Indexer](#).

Known Limitations

- To add a CVS repository, FishEye must have file system access to the repository. If you cannot install FishEye on the same server as CVS, then use [rsync to mirror](#) the repository.
- Currently, FishEye does not handle the \$Log RCS expansion keyword correctly. Some diff results (and line numbers in diffs) may appear incorrect in files where \$Log is used.

CVS repository setup

To add an external CVS repository to FishEye:

1. Click the 'cog' menu in the FishEye header, and choose **Administration** (you'll need to be logged in as an [administrator](#) to see this link).
2. Click **Repositories** (under 'Repository Settings').
3. Click **Add repository**.
4. Complete the wizard:

Repository Type	Choose CVS .
Name	A name for this repository. The name may contain alphanumeric, underscore, '-' or '.' characters. Use 'cvs' if you can't think of a better name.
Description	A short description of this repository.
CVS Dir	The path to the CVS repository. This is often /usr/local/cvsroot. This is a path in the server's file system.
Charset	The character set used to interpret and display text files.
Enable immediately	Controls whether FishEye will immediately enable this repository, which starts the initial scan. If you wish to do some further configuration before the scan starts, then select 'No'. You can enable a repository later from the Repository List.

Git

This page describes how to use FishEye's native repository access to connect to a Git repository. The process for doing this depends on where the Git repository is hosted – connecting to a Git repo that is hosted in [Atlassian Bitbucket Server](#) is much easier.

FishEye interacts with Git repositories by executing the Git command in a separate process. Hence, the server running FishEye needs to have Git installed. FishEye indexes Git repositories by making a private, bare clone of your repository within FishEye's cache area. It uses this private clone for most Git operations.

Requirements

- Please refer to [Supported platforms](#) to see the version of Git that is required by FishEye.
- You must have [set the path to the Git executable](#) in FishEye before you can add Git repositories.

On this page:

- [Requirements](#)
- [Git repositories hosted in Bitbucket Server](#)
- [Git repositories hosted elsewhere](#)
- [Limitations](#)

Related pages:

- [Authentication](#)
- [Git Client Configuration](#)
- [Crucible Repository Configuration](#)
- [General Crucible Configuration](#)

Git repositories hosted in Bitbucket Server

When Bitbucket Server is integrated with FishEye:

- You can easily add Git repositories to FishEye. The repository behaves just like a native repository in FishEye, so your team gets all the benefits of FishEye indexing, browsing and searching.
- The repository becomes available to Crucible (when integrated), so you can perform in-depth code reviews for changes in the repository.
- When you add a Bitbucket Server repository to FishEye, a push to the repository will by default automatically trigger FishEye to run an incremental index. No further configuration is required – you don't have to configure polling for new commits, or set up dedicated FishEye web hooks in your Bitbucket Server instance.

You'll need to have an account in the Bitbucket Server instance, as well as permission to view the repository that you want to add.

1. Click the 'cog' menu in the FishEye header, and choose **Administration** (you'll need to be logged in as an [administrator](#) to see this link).
2. Click **Repositories** (under 'Repository Settings').
3. Click the **Bitbucket repositories** tab, and authenticate with Bitbucket Server if necessary.
4. If multiple instances of Bitbucket Server are [connected](#) to FishEye, use the **Bitbucket server** list to choose the instance of Bitbucket Server that hosts the repository you wish to add.
5. Optionally, type a filter pattern to restrict the list of displayed repositories to those with a matching name, key or project.
6. Click **Add** for each repository that you wish to add to FishEye.

When you add a Bitbucket Server repository:

- FishEye creates a read-only SSH key and adds that as an [access key](#) to the repository in Bitbucket Server. If this operation fails, the key will be added as a user key to your profile in Bitbucket Server. You can check if an access key was added by viewing the repository's settings in Bitbucket Server. See [Using SSH keys to secure Git operations](#) in the Bitbucket Server documentation for more information.

Git repositories hosted elsewhere

When adding or managing a Git repository, you can do the following:

- Define repository details, as described below.
- Set FishEye's [repository options](#).

To add an external Git repository to FishEye:

1. Click the 'cog' menu in the FishEye header, and choose **Administration** (you'll need to be logged in as an [administrator](#) to see this link).
2. Click **Repositories** (under 'Repository Settings').
3. Click **Native repository access**, and then **Add repository**.
4. Complete the wizard:

Step 1

Field	Description
Repository Type	Select Git .
Name	Enter a name for this repository. The name may contain alphanumeric, underscore, '-' or '.' characters. This will be repository name in FishEye.
Description	Enter a short description of this repository.

Step 2

Field	Description
Repository Location	Enter the URL describing the Git repository location. FishEye will clone this repository for indexing purposes. You can use any URL recognized by Git itself. FishEye recognizes the following URL's: git://server_name[:port]/path_to_repository http://server_name[:port]/path_to_repository https://server_name[:port]/path_to_repository ssh://server_name[:port]/path_to_repository file://[hostname]/path_to_repository Do not use spaces in your URL.
Path	(<i>optional</i>) Enter the path within the Git repository that you want FishEye to index. This lets you limit FishEye to indexing a subset of the complete Git repository.
Authentication Style	Choose the desired authentication style for your repository — No authentication , Generate key pair for SSH , Upload private key for SSH or Password for http(s) . Please refer to Authentication for more information.
Block Size	(<i>optional</i>) Enter how many commits you want FishEye to process in one batch. Larger values require more memory and increase the amount of work FishEye commits to the database in a single operation. The default is 400. The minimum being 1. This field only accepts positive whole numbers. Requires a repository restart.
Command Timeout	(<i>optional</i>) Enter the time that a single Git command is allowed to take to execute. Any command that exceeds this time is terminated and the operation will fail.
Rename Detection	(<i>optional</i>) Select which Git rename detection strategy FishEye will use to detect copy and move operations within the repository. Please refer to the Git documentation for more information .

Step 3

Field	Description

Store Diff Info	Check this if you want FishEye to cache information about file diffs in its database. This is required for some FishEye features. See Configuring Repository Details for more information on this setting.
Enable Repository After Adding	Check this to enable the repository after adding (i.e. when you click the Add button).

Limitations

Indexing of a Git repository can be slow when new branches are pushed. Performance in such cases can be improved by setting a command line option for FishEye. See this [KB page](#) for more information.

Mercurial

When adding or managing a Mercurial repository, you can do the following:

- Define repository details, as described below.
- Set FishEye's [repository options](#).

Requirements

- Please refer to [Supported platforms](#) to see the version of Mercurial that is required by FishEye. (You should restart FishEye after upgrading Mercurial.)
- Before you add your Mercurial repositories, you will need to [set your executable location](#) for hg (Mercurial).

On this page:

- [Requirements](#)
- [Mercurial repository setup](#)
- [Notes](#)

Related pages:

- [Authentication](#)
- [Mercurial Client Configuration](#)
- [Configuring Crucible repositories](#)
- [General Crucible Configuration](#)

Mercurial repository setup

1. Click the 'cog' menu in the FishEye header, and choose **Administration** (you'll need to be logged in as an [administrator](#) to see this link).
2. Click **Repositories** (under 'Repository Settings').
3. Click **Add repository**.
4. Complete the wizard:

Step 1

Field	Description
Repository Type	Select Mercurial .
Name	Enter a name for this repository. The name may contain alphanumeric, underscore, '-' or '.' characters. This will be repository name in FishEye.
Description	Enter a short description of this repository.

Step 2

Field	Description
Repository Location	Enter the URL describing the Mercurial repository location. FishEye will clone this repository for indexing purposes.  This URL must be compatible with the hg clone command.
Authentication Style	Choose the desired authentication style for your repository — No authentication , Generate key pair for SSH , Upload private key for SSH or Password for http(s) . Please refer to Authentication for more information.
Block Size	(optional) Enter how many commits you want FishEye to process in one batch. Larger values require more memory and increase the amount of work FishEye commits to the database in a single operation. The default is 400. The minimum being 1. This field only accepts positive whole numbers.
Command Timeout	(optional) Enter the time that a single Mercurial command is allowed to take to execute. Any command that exceeds this time is terminated and the operation will fail.

Step 3

Field	Description
Store Diff Info	Tick this checkbox, if you want FishEye to cache information about file diffs in its database. This is required for some FishEye features. See Configuring Repository Details for more information on this setting.
Enable Repository After Adding	Tick this checkbox, to enable the repository after adding (i.e. when you click Add).

Notes

- **Submitting Feedback** — We're very interested in your feedback regarding this feature. The best place for submitting feedback is the [FishEye forums](#).

Perforce

When adding or managing a Perforce repository, you must:

- Define repository details, as described below.
- Set FishEye's [repository options](#).

Requirements

- Please refer to [Supported platforms](#) to see the version of Perforce that is required by FishEye.
- You must have [set the path to the Perforce executable](#) in FishEye before you can add Perforce repositories.

Perforce repository setup

1. Click the 'cog' menu in the FishEye header, and choose **Administration** (you'll need to be logged in as an [administrator](#) to see this link).
2. Click **Repositories** (under 'Repository Settings').
3. Click **Add repository**.
4. Complete the wizard:

Repository Type	Choose Perforce .

Name	A name for this repository. The name may contain alphanumeric, underscore, '-' or '.' characters. Use 'perforce' if you can't think of a better name.
Description	A short description of this repository.
Host	The name of the server which provides the Perforce repository.
Path	The path within the Perforce depot that you want FishEye to index. You would normally put the depot path here, e.g. //depot/ but you may also use a more specific path to restrict FishEye to a subset of the depot. The Perforce wildcards '*' and '...' are not supported.
Port	(Optional) The port the server is listening on. FishEye will default to the standard Perforce port (1666) if you do not specify a value here.
Username	The account that FishEye should use to connect to the Perforce repository.
Password	The password for the account that FishEye should use to connect to the Perforce repository.
Block size	Controls how many changelists FishEye will fetch from the depot in one batch. Larger values can reduce the time it takes for FishEye to scan your repository for changes, but use more memory. The default is 400. The minimum being 1. This field only accepts positive whole numbers.
Filelog limit	FishEye uses the P4 filelog command to gather information about the files in changesets. The list of files generated can be very large. Setting a limit here will cause FishEye to batch up filelog operations into groups. This is useful with some versions of the Perforce client which may have trouble with large output. In general you should only set this field if you have a 2005 client or earlier. Lower values will degrade scanning performance.
P4 Operation Timeout	Sets the timeout value that FishEye imposes on P4 operations. Operations which exceed this value are terminated. The default for most operations is 10 minutes. Use the following terms to specify particular units of time: s, mi, h, d, w, mo, y (for seconds, minutes, hours, days, weeks, months and years respectively). For example, 10s.
Throttle connections-per-sec	If set, this allows FishEye to throttle how many connections it makes per second to the Perforce server. The default is blank (do not throttle). You may enter fractional values such as 2.5.
Charset	The character set used to interpret and display text files.
Unicode Server	This field indicates whether the Perforce Server is running in internationalized mode.
Skip Labels	When true, FishEye will not scan Perforce Labels for FishEye tag information.
Case Sensitive	This field indicates whether the Perforce Server metadata is case sensitive. You should set this to 'false' for servers running on Windows platforms.
Disable Multiple Print	When FishEye needs file content from Perforce, it uses a p4 print operation. Normally FishEye will request multiple files in one operation but this can cause problems in some Perforce instances. Set this value to true to disable printing multiple files in one operation.
Start Revision	If you wish to set this, please enter a changelist number. If set, the revision number from which FishEye will start indexing the repository. The default is to start scanning from the first revision in the repository.

Initial Import	When a Start Revision is set, this setting controls how FishEye establishes the initial state of the repository. When true, FishEye will import the repository content as it existed one revision prior to the start revision. FishEye will create a single synthetic revision to hold the initial state. The comment associated with this revision will be 'Created by FishEye for initial repository import'. False means that FishEye will only process the revisions from the start revision onwards. The repository state prior to this revision is ignored.
Username/Password	The credentials to use if your repository requires authentication.
Store Diff Info	Enable this option if you are using the Subversion or Perforce SCM systems and making use of per-author line counts. Otherwise, enabling this option is not necessary. Read more information
Enable immediately	Controls whether FishEye will immediately enable this repository, which starts the initial scan. If you wish to do some further configuration before the scan starts, then select 'No'. You can enable a repository later from the Repository List.

Subversion

When adding or managing a SVN (Subversion) repository, you can:

- Define repository details, as described below.
- Set FishEye's [repository options](#).

There are also the following SVN-specific actions:

- [Updater](#)
- [Indexer](#)
- [Store diff info](#)

On this page:

- [Requirements](#)
- [SVN repository setup](#)
- [Finding your repository root](#)

Requirements

- Please refer to [Supported platforms](#) to see the version of Subversion that is required by FishEye.
- You may need to have set up a [Subversion client](#) in FishEye before you can add Subversion repositories.
- Grant [permission to FishEye](#) to scan your repository.
- Set up the correct [branch and tag structure](#).

It is particularly important that you set up the correct branch and tag structure for your Subversion repositories. If FishEye does not know which files are tags and branches, it will treat all files as trunk files. This can significantly increase the effective size of your repository. **This will increase initial scan time and impact runtime performance.** Please refer to the instructions on [tag and branch configuration](#).

In the majority of cases, indexing a small repository shouldn't take hours, and certainly not days. However, if you have a giant repository, have a slow remote host, you're using HTTP or HTTPS protocols, or if there is a problem with the symbolic setup of your repository, it could potentially take hours or even days. If in doubt, schedule the indexing to run over a weekend or extended maintenance period.

Using the 'file://' protocol to access your Subversion repository can be much faster than the other network protocols if your repository is on a local disk on the same server as FishEye.

Knowledge Base

You may find useful information in either the FishEye [Subversion FAQ](#) or the [Subversion integration](#) sect

ion of the FishEye Knowledge Base.

SVN repository setup

1. Click the 'cog' menu in the FishEye header, and choose **Administration** (you'll need to be logged in as an [administrator](#) to see this link).
2. Click **Repositories** (under 'Repository Settings').
3. Click **Add repository**.
4. Complete the wizard:

Screenshots: Adding a SVN repository

Add Repository - Page 1 of 3

Basic Details

Repository Type:

Name: *

Description:

Add Repository - Page 2 of 3

SVN Connection Details

SVN URL: *

Path:

Username:

Password:

Add Repository - Page 3 of 3

Final Settings

Store Diff Info: ☒

Enable Repository After Adding: ☐

Setting	Description
Repository Type	Choose Subversion .
Name	A name for this repository. The name may contain alphanumeric, underscore, '-' or '.' characters. Use 'svn' if you can't think of a better name.
Description	A short description of this repository.
SVN URL	The Subversion Repository Root URL to your repository, such as svn://svn.foo.com or file:///var/svn . If you are not sure what your repository root is, check the section below "Finding your Repository Root". (Please note that file protocol performs the fastest followed by svn and lastly by http/s. Therefore where possible please use the file protocol.)

Path	The sub-tree within your repository that FishEye should display. If this value is '.' (or empty), then the whole repository will be shown.
Username/Password	The credentials to use if your repository requires authentication. See also SVN Authentication Issues .
Store Diff Info	Enable this option if you are using the Subversion or Perforce SCM systems and making use of per-author line counts. Otherwise, enabling this option is not necessary. Read more information
Enable Repository After Adding	Controls whether FishEye will immediately enable this repository, which starts the initial scan. If you wish to do some further configuration before the scan starts, then do not check. You can enable a repository later from the repository list.

Once the repository is added you can click the repository name on the '[Repositories](#)' screen to configure the following settings:

Setting	Description
Block Size	Controls how many revisions FishEye will pull down from the repository in one batch. Larger values can reduce the time it takes for FishEye to scan your repository for changes, but use more memory. Smaller values can reduce the amount of memory FishEye uses during scans. The default is 400. The minimum being 1. This field only accepts positive whole numbers. Requires a repository restart.
SVN Operation Timeout	Sets the timeout value that FishEye imposes on Subversion operations. Operations which exceed this value are terminated. The default for operations is 1 hour. It can be changed to a different interval, for example: "2 days", "10 hours", or "20 minutes". The library that FishEye/Crucible uses (SVNKit) for SVN integration defines a timeout of 60 minutes by default if the property <code>http-timeout</code> is not defined in the Subversion configuration. This property defines the time to wait for a server response and terminates the HTTP call if it exceeds the timeout. See this KB for more information: SVN operations taking longer than an hour time out
Throttle connections-per-sec	If set, this allows FishEye to throttle how many connections it makes per second to the SVN server. Many systems use <code>inetd/xinetd</code> to service the <code>svnserve</code> protocol. <code>xinetd</code> has, by default, an incoming connection limit which can cause FishEye to disrupt other <code>svnserve</code> -based connections. The default is blank (do not throttle).
Charset	The character set used to interpret and display text files.
Access Code	The access code for the <code>fisheye.access</code> property on the server. See also Subversion fisheye.access .
MD5 Access Code	The MD5 sum of the above Access Code. See also Subversion fisheye.access . (This field only appears if Access Code is set.)
Set Access Property Command	The Subversion command to set the <code>fisheye.access</code> property to grant FishEye access if necessary. See also Subversion fisheye.access . (This field only appears if Access Code is set.)
Start Revision	If set, the revision number from which FishEye will start indexing the repository. The default is to start scanning from the first revision in the repository.

Initial Import	When a Start Revision is set, this setting controls how FishEye establishes the initial state of the repository. Do not import means that FishEye will only process the revisions from the start revision onwards. The repository state prior to this revision is ignored. Import without tag information means that FishEye will import the repository content as it existed one revision prior to the start revision. FishEye will create a single synthetic revision to hold the initial state. The comment associated with this revision will be 'Created by FishEye for initial repository import'. Tags created prior to the start revision are ignored. Due to the synthetic commit the "Last commit" date will display the date "31 Dec 69" in statistics panels.
Follow Base Moves	If you have changed the location of your repository in your Subversion server over time, set this option to true. Doing so will cause FishEye to index your repository's full history across all its locations within your server.
Use Built-in Symbolic Rules	Determines how FishEye attempts to understand the tag and branch structure of your Subversion repository. Read more information.

If FishEye returns the error 'Authentication Canceled', this may simply indicate an incorrect username or password.

Finding your repository root

Run the following command:

```
svn info SVN_URL
```

Where SVN_URL is the complete URL of the repository you want to add.

You will get something like the following:

```
>svn info http://svn.example.com/svn5/

Path: svn5
URL: http://svn.example.com/svn5/
Repository Root: http://svn.example.com/
Repository UUID: ce062a09-193b-427a-a7b3-a85007076e5d
Revision: 83
Node Kind: directory
Last Changed Author: ryan
Last Changed Rev: 83
Last Changed Date: 2009-05-07 10:48:41 +1000 (Thu, 07 May 2009)
```

Next to "Repository Root" is the URL you should define as your repository root. The path will be whatever is remaining.

SVN fisheye.access

The `fisheye.access` property allows an administrator/commmitter to control FishEye access to a directory in the repository. FishEye queries this property to decide whether it will continue to access the repository. If the property exists, but does not match that configured in FishEye, FishEye will immediately disconnect from the repository.

By default, FishEye will have access to your repository and you do not need to configure this access mode. It is only necessary if you identify a need to restrict FishEye access to your repository (this is

generally not required).

Setting FishEye Access Mode

FishEye can operate in one of three access modes:

Mode	Access	Subversion repository property: <code>fisheye.access</code>
Allow	Any FishEye server	'allow' or no property set
Access Code	Only FishEye servers configured with the correct Access Code	e.g. 'md5:dc0c08df1f3e80b599c90f53d7dd05ec'
Deny	No FishEye server	'deny'

If you would like to restrict FishEye access to your repository, you must set the `fisheye.access` property. This property must be set on the 'URL + path' you have configured in FishEye.

Setting an Access Code

The repository must be configured with the MD5 sum of the Access Code that is configured in FishEye. The MD5 sum and even the `svn` command to set the property will be generated for you by FishEye when you configure the repository using the FishEye Administration page. See [Subversion repository details](#).

For example, if you have configured FishEye with a URL of `svn://foo.com/`, a path of `.` and an Access Code of 'fisheye', then you would need to do something like this:

```
$ svn checkout -N svn://foo.com/ tmpworkspace
$ cd tmpworkspace
$ svn propset fisheye.access "md5:4d0c5db8382f80c58e7b0619ae5767a7" .
$ svn commit -m "grant fisheye access"
```

Denying Access to all FishEye Instances

To deny all FishEye instances access to the repository, it must be configured with the `fisheye.access` property of 'deny'.

For example, if you have configured FishEye with a URL of `svn://foo.com/` and a path of `.` (or you have left path empty), then you would need to do something like this:

```
$ svn checkout -N svn://foo.com/ tmpworkspace
$ cd tmpworkspace
$ svn propset fisheye.access "deny" .
$ svn commit -m "disable fisheye access"
```

If you configured a path of `some/dir` then use:

```
$ svn checkout -N svn://foo.com/some/dir tmpworkspace
$ cd tmpworkspace
$ svn propset fisheye.access "deny" .
$ svn commit -m "disable fisheye access"
```

SVN tag and branch structure

This page provides a brief explanation of how Subversion works, how FishEye interacts with it, and examples of how to configure FishEye to work with Subversion according to your needs.

In a Subversion repository, branches and tags can be easily copied or duplicated — this is done by creating a form of pointer or reference from one location to another, avoiding the need to duplicate a lot of information. The disadvantage of this is that Subversion repositories can be confusing to administer at times and its internal complexity can be problematic for applications such as FishEye that need to finely process its contents. As a result, FishEye may require some in-depth configuration with Subversion.

On this page:

- [FishEye automatic presets](#)
- [Introduction to Subversion conventions](#)
- [Choosing the correct configuration method](#)
- [Custom layouts](#)
- [Interpreting multiple rules](#)
- [Examples](#)
 - [Ideal configuration example](#)
 - [Custom example 1](#)
 - [Custom example 2](#)
 - [Example from a FishEye customer](#)
- [How Subversion works](#)
- [Frequently Asked Questions](#)

FishEye automatic presets

FishEye provides automatic presets that match those defined in the Subversion Red Book (at <http://svnbook.red-bean.com/en/1.1/ch04s07.html#svn-ch-4-sect-7.1>).

If your repository structure strictly follows either of the Subversion conventions then these presets will be suitable.

To apply a preset, go to the repository configuration page (click the repository name on the ['Repositories'](#) screen) and choose a preset from the **And then Apply the Following Rules** list.

However if you have defined your svn structure via the custom symbolic rules, then you should set "Use In-built Symbolic Rules" to no.

The **Use Built-in Symbolic Rules** checkbox applies another regex which does a deep search for trunk, tag and branch directories. If your structure exactly matches the selected symbolic rule set, then it is safe to uncheck this.

Note that the preset rules are case-sensitive – if your SVN folder structure doesn't use all lowercase naming then you will need to define your own symbolic rules based on the defaults.

See [Interpreting multiple rules](#) below.

SVN Symbolic Rules

Use Built-in Symbolic Rules:

None

/trunk/..., /branches/NAME/..., /tags/NAME/...

✓ /project/trunk/..., /project/branches/NAME/..., /project/tags/NAME/...

Custom

And then Apply the Following Rules:

Customise

	Regular Expression	Name	Logical Path Prefix
Trunk	(((^/)+)/trunk(/ \$))	trunk	/\${1}
Branches	(((^/)+)/branches/NAME(/ \$))	branches/NAME	/\${1}/\${2}
Tags	(((^/)+)/tags/NAME(/ \$))	tags/NAME	/\${1}/\${2}

Introduction to Subversion conventions

The most common repository structure in Subversion is to have trunk, branches and tags directories for each

project, like so:

```
/project1/trunk/...  
/project1/branches/b1  
/project1/branches/b2  
/project1/branches/...  
  
/project1/tags/t1  
/project1/tags/t2  
/project1/tags/...
```

If you point your FishEye repository at a single project root, say `/project1/`, then to FishEye, the directory structure looks like this:

```
/trunk  
/branches  
/tags
```

In this case, you have one single project within the repository root. On the other hand, if you point FishEye at the repository root, you will have multiple projects visible in your FishEye view.

i Note that this example follows common conventions for repository layout, as outlined in the [Subversion Red Book](#). If you follow these conventions, your FishEye configuration will be simpler.

Choosing the correct configuration method

If you are unsure as to whether you can use a preset symbolic rule, or whether you need to write your own custom symbolic rules, refer to the following:

1. If you point FishEye at your project root:

```
/PROJECT1
```

and you have the following structure at this level of the repository:

```
trunk  
branches  
tags
```

Then use FishEye's built-in rules for easy, one-step configuration.

2. If you point FishEye at the repository root, you have multiple projects and each project has the following structure:

```
trunk  
branches  
tags
```

leading to this structure in each project:

```

/PROJECT1/trunk
/PROJECT1/branches
/PROJECT1/tags

```

```

/PROJECT2/trunk
/PROJECT2/branches
/PROJECT2/tags

```

(and so on), then use FishEye's built-in rules for easy, one-step configuration.

3. Otherwise, you have a [custom layout](#).

Custom layouts

[Regular expressions](#) are required knowledge for creating custom repository layouts in FishEye.

If you have a custom repository structure, that is your repository structure does not follow [SVN conventions](#), you need to configure FishEye to recognize the paths in your repository. What you are telling FishEye is which paths within the repository are related, i.e. which are operations on the same file in different branches and which are tags of a file. You must also tell FishEye how to determine the branch name or the tag name. Most custom layouts are variations on the one of the two standard layouts described above. The best approach to creating your custom configuration is to use one of the appropriate entries from the drop down list. This can serve as a template for you, which you can then customize. Once you have selected the appropriate template, select the "Custom" entry from the drop down list. Now, you will be able to edit the entries (or add entries).

See the [examples](#) below for more information.

How to set a custom layout

Using [Regular Expressions](#), you can describe any custom tag or branch structure that you have. You should use one of the common layouts (from the drop down list) as a basis, firstly select it, then select 'Custom' to edit or add rules.

When looking at a file on a branch, or a file that was tagged, FishEye needs to determine a name for the branch or tag. FishEye does this by matching a regular expression against the file's path, and extracting the name based upon the match. FishEye also needs a name for files on the trunk. In effect, this is the name of the trunk 'branch'.

For any file that matches a trunk/branch/tag regular expression, a logical path is calculated. Two different files with the same logical path are considered to be related. For example, using the second type of common repository layout:

- The file `project1/trunk/dir1/foo.txt` would have a logical path of `project1/dir1/foo.txt`.
- The file `project1/tags/BUILD123/dir1/foo.txt` would have a logical path of `project1/dir1/foo.txt` and the name of the tag would be `project1-BUILD123`.
- Both these files have the same logical path, and so are considered related. By looking at the revision where the directory-copy for `project1/tags/BUILD123/dir1/foo.txt` occurred, FishEye can determine to what revision the tag `project1-BUILD123` applies.

You can add as many rules as you need. See [Interpreting multiple rules](#) below.

Rule	Description
Regex	The regular expression used to match against the start of the path. The trailing part of the path that does not match the regex is called the tail.

Name	An expression used to extract a tag or branch name from the regex.
Logical Path Prefix	This is an expression used to construct the logical path. The logical path is the concatenation of the result of this expression, and the tail of the regex.

See the [examples](#) below for more information.

Interpreting multiple rules

See also [Custom layouts](#) above.

To find which rule to apply, FishEye creates three sets of rules and tries them in this order: branches, tags, trunk.

For each set of rules, it finds the closest match within that set. If any rule in the set matches, FishEye will not try the next set. If multiple rules within a set match then FishEye will use the best match. The best match is the rule with the smallest logical tail (the trailing part of the path that does not match the regex).

Examples

These examples show the regular expressions used for some custom configurations. If you need more information on how these examples work, please see [SVN tag and branch structure](#) on this page.

- [Ideal configuration example](#)
This shows a best-case near "zero configuration" project structure that is instantly compatible with FishEye.
In this case, you have trunk, branches and tags as the base folders in your repository.
- [Custom example 1](#)
This shows a custom project structure and the symbolic rules required.
- [Custom example 2](#)
This shows another kind of custom project structure and the symbolic rules required.
- [FishEye customer example](#).
This is a real-world configuration used by a FishEye customer.

Ideal configuration example

If your repository is organized in this way, simply select the **In-Built symbolic rules** option. FishEye will then be fully connected to your repository (you do not need to write a regular expression, or choose anything from a list).

Project structure

```
/PROJECT1/trunk/

/PROJECT1/branches/branchname

/PROJECT1/tags/tagname
```

 Note that this example follows common conventions, as outlined in the [Subversion Red Book](#).

Custom example 1

Whenever you have a custom project structure in Subversion, you will need to write a [regular expression](#)

Say you have an additional directory you use for tagging releases, which is different from the everyday tags you create in the tags directory:

Project structure

```

/trunk/

/branches/branchname

/tags/tagname

/releases/releasename

```

Symbolic rules

Regular Expression	Name	Logical Path Prefix
<code>trunk(/ \$)</code>	<code>trunk</code>	N/A
<code>branches/([^/]+)</code>	<code>\${1}</code>	N/A
<code>(tags releases)/([^/]+)</code>	<code>\${2}</code>	N/A

Custom example 2

Whenever you have a custom project structure in Subversion, you will need to write a [regular expression](#).

In this example, there is a "core" project area and then a number of separate plugins. the core contains its own trunk/branches/tags structure while the plugins are in a named directory which contains their trunk/branches/tags directory. We want to have the core and all the plugins visible in a single FishEye repository.

Project structure

```

/core/trunk/
/core/branches/
/core/tags/

/plugins/plugin1/trunk/
/plugins/plugin1/branches/
/plugins/plugin1/tags/

/plugins/plugin2/trunk/
/plugins/plugin2/branches/
/plugins/plugin2/tags/

```

Symbolic rules

Applies to	Regular Expression	Name	Logical Path Prefix
trunk	core/trunk(/ \$)	trunk	core
trunk	plugins/([^\s/]+)/trunk(/ \$)	trunk	plugins
branches	core/branches/([^\s/]+)	core-\${1}	core
branches	plugins/([^\s/]+)/branches/([^\s/]+)	\${1}-\${2}	plugins
tags	core/tags/([^\s/]+)	core-\${1}	core
tags	plugins/([^\s/]+)/tags/([^\s/]+)	\${1}-\${2}	plugins

In this example, the Logical Path Prefix has been configured to distinguish files with the same name in different plugins. For example, the file build.xml may exist in all plugins but such files are not related even though they have the same name. The Logical Path Prefix is used to tell FishEye to which "logical group" the files belong.

Example from a FishEye customer

This is a real-world example from a FishEye customer. This is a slightly non-standard project structure. The correct symbolic rules for this project structure are shown below:

Project structure

```

/trunk/PROJECT1

/branches/PROJECT1/branchname

/tags/PROJECT1/tagname

```

Symbolic rules

Regular Expression	Name	Logical Path Prefix
trunk/([^\s/]+)	\${1}	N/A

<code>branches/([^\s]+)/([^\s]+)</code>	<code>\${1}-\${2}</code>	N/A
<code>tags/([^\s]+)/([^\s]+)</code>	<code>\${1}-\${2}</code>	N/A

How Subversion works

Since tags and branches are implemented via directory copies in Subversion, they are not really first-class concepts. This means that FishEye has to determine branch and tag information by examining the paths involved in Subversion operations and matching these against branch and tag conventions used in the repository. Since these conventions are not fixed, you may need to tell FishEye what conventions you use in your repository. By default FishEye has some inbuilt rules which handle the most common conventions typically used in most Subversion sites. If, however, you've decided to use a custom convention, you can define custom rules to describe what your tag/branch structure looks like. These settings can be edited on the 'Add Repository' or 'Edit Repository' pages in the FishEye Administration pages.

The symbolic setup tells FishEye how to classify each path it encounters as it indexes the repository. Each path is classified as either a trunk, branch, tag or root path. The trunk, branch and tag categories are the normal conventions used in SCMs. The root category is used when a path does not match any of the given trunk/branch/tag settings and is mostly treated in the same way as trunk paths. For example, the branches directory itself does not belong to the trunk, a particular branch or a tag and is classified as a root path.

The symbolic settings do not exclude any paths from consideration by FishEye. To exclude paths you should set up appropriate 'allow' rules. If your symbolic setup does not match a path, that path will be classified as a root path and processed by FishEye accordingly.

If you change these trunk/branch/tag settings, you would normally perform a complete re-scan of the repository to ensure FishEye's index is consistent with the settings. FishEye will suggest this when you make changes and you can also do this manually from the [Indexer](#) option. If you don't want to re-index, you can also choose to ignore this suggestion.

For more information on tag and branch layout, see [Repository Layout](#) in the Subversion documentation, or [How tags work in Subversion](#) for more background information.

Frequently Asked Questions

What is the logical path prefix?

The first part of a path, where you want to constrain to a specific origin for the paths in context.

What is the logical tail?

The last part of the path is the logical tail, that specifies the exact folders in context.

How does the logical tail affect the search?

The logical tail could be used to select every occurrence of a file called `build.xml`, for example, or every folder of a given name near the end of the path.

How tags work in Subversion

If you are having trouble finding out the correct Tag or Branch names to use under Subversion (especially when searching or creating EyeQL queries), the steps on this page will help.

Discovering your Subversion tag and branch structure

1. Open a file in your repository using FishEye. Look for a file that has a long lifetime (for example, your build script).
2. Look at the 'Tags' section to see examples of tags in your system. There are a few different conventions, but the tags shown will give you clues as to how your repository custodians do things.

A typical tag looks like this:

```
fisheye-build-82
```

3. Copy the tag(s) you may need from the examples you can see in FishEye's view.
4. Enter the tags exactly as they appear into your EyeQL queries or (FishEye Search).
5. Test the outcome.

Screenshot: Subversion Tags viewed in FishEye

Branch fisheye-FE_SVN_BRANCH	
6267	annotated / raw Diffs: previous, other Lines: 55 (+0, -11) Created: 2005-06-20 01:27:05 +1000 (2 years 7 months ago) Author: conor Changeset: 6267 NONE: Handle files with no logical path. Pass config around to have access to username/password info Tags: fisheye-build-82 fisheye-build-87 fisheye-build-88 fisheye-build-90 fisheye-build-91 Properties svn:eol-style = native svn:executable = * svn:keywords = Author Date Id Revision

How tag names are constructed

Hyphens are the default method for separating the tag elements, but tags in your FishEye instance may be different.

Your tag structure depends on several things:

- The symbolic structure of your repository
- The way your FishEye instance was set up
- Your organization's convention for naming things in the repository
- The configurable character that separates parts of the tag name.

This can sometimes make it confusing to guess what the tag structure could be, when you are searching in FishEye or using an EyeQL query. Additionally, the separator between tag elements can be configured as hyphen, colon, or other punctuation marks.

Note that slashes cannot be used in a tag name – these are converted to colons by default. For example if your symbolic setup would give you a tag name like the following:

```
path/project/fisheye-build-82
```

then your tag should look like this:

```
path:project: fisheye-build-82
```

For more information, see [SVN tag and branch structure](#).

Verifying SVN symbolic rules

This page describes a simple command-line tool ('svnrules') that you can use to find branch and tag creation changes for which there may not be SVN symbolic rules defined. The tool is available on all supported platforms but the examples below feature a Linux installation.

You may have [configured SVN symbolic rules](#) in FishEye if your SVN repository structure doesn't strictly follow the Subversion [Repository Layout](#) conventions; these rules allow FishEye to understand your custom layout.

A possible consequence of not using symbolic rules for custom repository layouts, or of configuring these incorrectly, is that FishEye may not recognize separate trunk, branch and tag creation changes, and spend excessive time indexing those as genuine changes.

The svnrules tool ships with FishEye 3.0, and later versions. It parses the SVN log and compares that against the SVN symbolic rules configured in FishEye to find commits that are likely to be branch or tag creation changes for which there are no SVN symbolic rule defined.

You may find the `svnrules` tool helpful if you use a custom SVN repository structure, and

- you want to analyze your repositories before indexing them, to avoid excessive time spent indexing
- you wish to analyze your repository structure yourself that must remain confidential.

Disclaimer: Please note that `svnrules` is very simple and doesn't attempt to perform a thorough inspection of the repository. It simply scans the repository history for copy/move operations, which in most cases indicate branch or tag creation changes. Those may be genuine copy operations within a repository, so false positives may be reported.

On this page:

- [Running the `svnrules` tool](#)
- [Interpreting the `svnrules` report](#)
- [Implementation information](#)

Related pages:

- [Subversion](#)
- [SVN tag and branch structure](#)
- [Troubleshooting SVN indexing issues in FishEye and Crucible](#)

Running the `svnrules` tool

The `svnrules` tool ships with FishEye and can be run from the command line as follows:

```
$ ./bin/fisheyectl.sh svnrules <REPOSITORY_NAME>
# e.g.
$ ./bin/fisheyectl.sh svnrules supercollider
```

You can point it to your existing FishEye instance folder with:

```
$ FISHEYE_INST=<FISHEYE_INST_FOLDER> ./bin/fisheyectl.sh svnrules
<REPOSITORY_NAME>
# e.g.
$ FISHEYE_INST=/home/fecru/fecru_inst ./bin/fisheyectl.sh svnrules
supercollider
```

The tool can be safely run when FishEye is already running, because it never attempts to modify any FishEye data or configuration files. It only analyzes the given repository structure and reports copy operations that may be unrecognized tag/branch creation changes because of missing symbolic rules.

If you run the tool on a properly configured repository that has no copy operations in its history, it would generate output similar to:

```
$ FISHEYE_INST=/home/fecru/fecru_inst ./bin/fisheyectl.sh svnrules
sanity-svn
INFO - Using log4j configuration file:
/home/fecru/fecru_home/log4j-client.xml
INFO - FishEye SVN Symbolic Rules Verifier
INFO - FishEye arguments: [sanity-svn]
INFO - loading config from file: /home/fecru/fecru_inst/config.xml
INFO - Testing connection to repo sanity-svn
INFO - Connected to repo sanity-svn
INFO - Processing svn log for a revision range: 0 ... HEAD
INFO - Verification complete, no issues found.
```

Any suspected copy operations are reported similarly to this:

```
$ FISHEYE_INST=/home/fecru/fecru_inst ./bin/fisheyectl.sh svnrules
supercollider
...
WARN - Copy operation recognised as commit to trunk, if this is branch
or tag creation change consider adding symbolic rule. Revision: 9678,
path: /trunk/xtralibs/iphone/libsndfile_iphone.a
WARN - Copy operation recognised as commit to trunk, if this is branch
or tag creation change consider adding symbolic rule. Revision: 9748,
path: /trunk/build/SCClassLibrary/JITLib/basics/EnvironmentRedirect.sc
WARN - Copy operation recognised as commit to trunk, if this is branch
or tag creation change consider adding symbolic rule. Revision: 9821,
path: /trunk/iphone/lib
WARN - Copy operation recognised as commit to trunk, if this is branch
or tag creation change consider adding symbolic rule. Revision: 9822,
path: /trunk/mac/Standalone Resources
WARN - Copy operation recognised as commit to trunk, if this is branch
or tag creation change consider adding symbolic rule. Revision: 9871,
path: /trunk/common/Packager
WARN - Copy operation recognised as unknown commit, if this is branch
or tag creation change consider adding symbolic rule. Revision: 9901,
path: /packages/ubuntu/supercollider-vim.menu
WARN - Copy operation recognised as unknown commit, if this is branch
or tag creation change consider adding symbolic rule. Revision: 9901,
path: /packages/ubuntu/supercollider-emacs.menu
WARN - Copy operation recognised as commit to trunk, if this is branch
or tag creation change consider adding symbolic rule. Revision: 10045,
path:
/trunk/common/build/SCClassLibrary/Common/GUI/osx/scide_scapp/viewExtens
ionsOSX.sc
WARN - Copy operation recognised as commit to trunk, if this is branch
or tag creation change consider adding symbolic rule. Revision: 10047,
path: /trunk/common/build/SCClassLibrary/Common/GUI/SCViewHolder.sc
WARN - Copy operation recognised as commit to trunk, if this is branch
or tag creation change consider adding symbolic rule. Revision: 10048,
path:
/trunk/common/build/SCClassLibrary/Common/GUI/guicrucial/StringGui.sc
WARN - Copy operation recognised as commit to trunk, if this is branch
or tag creation change consider adding symbolic rule. Revision: 10058,
path: /trunk/common/build/SCClassLibrary/JITLib/GUI/JITGui.sc
WARN - Copy operation recognised as commit to trunk, if this is branch
or tag creation change consider adding symbolic rule. Revision: 10059,
path: /trunk/common/build/Help/Libraries/JITLib/GUI
WARN - Copy operation recognised as commit to trunk, if this is branch
or tag creation change consider adding symbolic rule. Revision: 10069,
path:
/trunk/common/build/SCClassLibrary/Common/Collections/EnvironmentRedirec
t.sc
WARN - Copy operation recognised as commit to trunk, if this is branch
or tag creation change consider adding symbolic rule. Revision: 10070,
path: /trunk/common/build/SCClassLibrary/Common/Streams/History.sc
WARN - Copy operation recognised as commit to trunk, if this is branch
or tag creation change consider adding symbolic rule. Revision: 10072,
path: /trunk/common/build/SCClassLibrary/JITLib/GUI/ProxyMixerOld.sc
```

```
WARN - Copy operation recognised as commit to trunk, if this is branch
or tag creation change consider adding symbolic rule. Revision: 10124,
path: /trunk/common/Source/common/iPhone/FileBrowserViewController.mm
WARN - Copy operation recognised as commit to trunk, if this is branch
or tag creation change consider adding symbolic rule. Revision: 10214,
path: /trunk/common/include/nova-simd/vec.hpp
WARN - Copy operation recognised as commit to trunk, if this is branch
or tag creation change consider adding symbolic rule. Revision: 10243,
path: /trunk/common/cmake_modules/FindFFTW3f.cmake
WARN - Verification complete, 76 issues found.
WARN - Most common issues:
WARN - Path /trunk reported 72 times.
WARN - Path /trunk/build reported 36 times.
WARN - Path /trunk/build/Help reported 22 times.
WARN - Path /trunk/common reported 12 times.
WARN - Path /trunk/build/SCClassLibrary reported 11 times.
WARN - Path /trunk/build/SCClassLibrary/Common reported 8 times.
WARN - Path /trunk/common/build reported 8 times.
```

```
WARN - Path /trunk/common/build/SCClassLibrary reported 7 times.  
WARN - Path /trunk/common/build/SCClassLibrary/Common reported 5 times.  
WARN - Path /trunk/build/Help/GUI reported 5 times.
```

As you can see, there are number of warnings reported, followed by a summary of the most common issues.

Interpreting the svnrules report

Symbolic rule evaluation outcome	Issue reported
path recognized to be branch or tag	No issue reported – seems we have the correct SVN symbolic rule in place
path recognized as a trunk	<i>Copy operation recognized as commit to trunk, if this is branch or tag creation change consider adding symbolic rule.</i> False positives would be reported here where file(s) or folder(s) were genuinely copied within a trunk.
path not recognized as trunk, branch or tag	<i>Copy operation recognized as unknown commit, if this is branch or tag creation change consider adding symbolic rule.</i> You may want to advise the user to consider excluding this folder from indexing.

Implementation information

Internally, the tool runs the following command on the selected repository:

```
svn log -v
```

For large repositories this may take a significant amount of time. You may want to analyze parts of the history, using the -s and -e options:


```

$ ./bin/fisheyectl.sh svnrules --help
INFO - Using log4j configuration file:
/home/fecru/fecru_home/log4j-client.xml
INFO - FishEye SVN Symbolic Rules Verifier
INFO - FishEye arguments: [--help]
ERROR - Usage: fisheyectl.sh svnrules repname
Verifies symbolic rules on selected svn repository.
Please note this command may take very long time to run.
OPTIONS:
    -s, --start          start analysis from given revision number
                        (numeric or START), default START
                        where START is start revision configured in
                        Fisheye for selected repository
                        or revision 1 if start revision is empty.
    -e, --end            finish analysis on a given revision number
                        (numeric or HEAD), default HEAD
    -h, --help           this help

# e.g. to analyse only changesets 5..10 use:
$ FISHEYE_INST=/home/fecru/fecru_inst ./bin/fisheyectl.sh svnrules -s 5
-e 10 sanity-svn
INFO - Using log4j configuration file:
/home/fecru/fecru_home/log4j-client.xml
INFO - FishEye SVN Symbolic Rules Verifier
INFO - FishEye arguments: [sanity-svn]
INFO - loading config from file: /home/fecru/fecru_inst/config.xml
INFO - Testing connection to repo sanity-svn
INFO - Connected to repo sanity-svn
INFO - Processing svn log for a revision range: 5 ... 10
INFO - Verification complete, no issues found.

```

Setting up a repository client

- [CVS client](#)
- [Git client](#)
- [Mercurial client](#)
- [Perforce client](#)
- [Subversion client](#)

CVS client

FishEye supports CVS repository access, but this does not require installation of a separate client.

As FishEye reads your CVS repository directly from the local file system, using a CVS repository is a 'zero-installation' scenario. Once you have set up FishEye on the system where your CVS repository resides, FishEye should be able to access it automatically once you have configured it.

See [CVS](#) for more information about how to add a CVS repository to FishEye.

Git client

You need to set the path to the Git client executable in FishEye, as described on this page, before you can [add Git repositories](#) to FishEye.

Please see the [Supported Platforms](#) page for information about the versions of Git that are supported by FishEye.

Configuration

To set the location of the Git client:

1. Go to the [admin area](#) in FishEye.
2. Click **Server** (under 'Global Settings').
3. Click **Edit Details** (under 'Git Executable').
4. Enter the path to your Git binary executable. At present, you need to have the Git executable in the path of the user used to run your FishEye instance.
5. Click **Update**.
6. Restart your FishEye server.

Related Links

- [Git Configuration](#)
- [Crucible Repository Configuration](#)
- [General Crucible Configuration](#)

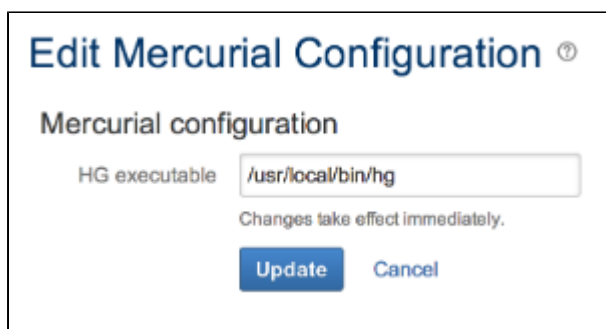
Mercurial client

You need to set the path to the Mercurial client executable in FishEye, as described on this page, before you can [add Mercurial repositories](#) to FishEye.

Please see the [Supported Platforms](#) page for information about the versions of Mercurial that are supported by FishEye.

Set the location of the Mercurial client

1. Go to the [admin area](#) in FishEye.
2. Click **Server** (under 'Global Settings').
3. Click **Edit Details** (under 'Hg Executable').
4. Enter the path to your hg (Mercurial) binary executable. If you followed the standard installation procedure, the default location is /opt/local/bin or /usr/local/bin for Linux and Mac OS X, and /Program Files/Mercurial for Windows.
5. Click **Update**.
6. Restart your FishEye server.



Edit Mercurial Configuration ⓘ

Mercurial configuration

HG executable

Changes take effect immediately.

Update **Cancel**

Related pages

- [Mercurial Configuration](#)
- [Crucible Repository Configuration](#)
- [General Crucible Configuration](#)

Perforce client

FishEye can communicate with any Perforce *server*, but it needs to use the P4 command-line client to do so.

By default, FishEye looks for the P4 executable in the current path. However, if necessary you can set the exact path to the P4 client executable in FishEye, as described on this page, before you [add Perforce repositories](#) to FishEye.

Please see the [Supported Platforms](#) page for information about the versions of Perforce that are supported by FishEye.

Configuration

To set the location of the Perforce client:

1. Go to the [admin area](#) in FishEye.
2. Click **Server** (under 'Global Settings').
3. Click **Edit Details** (under 'Perforce client').
4. Enter the path to your P4 (Perforce) binary executable.
5. Click **Update**.
6. Restart your FishEye server.

Files incorrectly considered binary

Some users have reported errors where FishEye considers some files to be binary when they are not. It appears this may be a limitation of earlier P4 clients. If you can upgrade to a recent P4 client (2006.1 onwards), this will fix this issue. You do not need to update the P4 Server. Please note that the repository will need to be reindexed in FishEye after upgrading the P4 client for the changes to be picked up.

If you are unable to upgrade to a recent P4 client, the [Repository Details](#) page in FishEye allows you to set a limit on the size of filelog commands sent to the server. Setting this to something around 100 will fix the issue. It will, however, also impact performance significantly.

Subversion client

FishEye can communicate with any Subversion *server* running version 1.1 or later, but it needs to use a Subversion *client* to do so. You can *not* [add Subversion repositories](#) to FishEye unless FishEye can access a Subversion client.

The SVNKit client is included in the current FishEye package (version 1.4.2 onwards). This is the default client for interfacing with Subversion, is generally the easiest to use, requires "zero-installation" and will be used automatically unless another client is specified. The SVNKit client is recommended for most users.

The alternative is the native client, which should only be used if the SVNKit client is unsuitable. See the configuration instructions below.

Please see [Native support for SVN](#) for information about FishEye native client compatibility.

Please see the [Supported Platforms](#) page for information about the versions of Subversion clients that are supported by FishEye.

Related pages:

- [SVNKit Client](#)
- [Native Subversion Client](#)
- [Native support for SVN](#)

Using the 'file://' protocol to access your Subversion repository can be much faster than the other network protocols. We recommend using the 'file://' protocol if possible.

Using Subversion 1.3 or later is strongly recommended. Versions prior to 1.3 are no longer supported by the Subversion project. They will work with FishEye, but you may want to consider upgrading to a supported version.

Native Subversion Client

FishEye can use a native Subversion client installed on your system, but your client **must include the JavaHL bindings**. FishEye can use all of the protocols supported by your native client.

Please see [Native support for SVN](#) for information about FishEye native client compatibility.

Please see the [Supported Platforms](#) page for information about the versions of Subversion clients that are supported by FishEye.

Getting native client libraries

Pre-compiled native clients are available for most platforms. The [Subversion download page](#) links to platform specific distributions. **Ensure you get the binary that includes JavaHL bindings**, as well as the standard package. Also ensure that the versions of the JavaHL and standard packages match.

Windows

To install Subversion for Windows, visit [this page](#).

You need to download the standard package as well as the **JavaHL** version. The standard package is named **svn-X.Y.Z-setup.exe** and the JavaHL installer file is named **svn-win32-X.Y.Z_javahl.zip** where 'X.Y.Z' refers to the version number (for example, `svn-win32-1.4.6_javahl.zip` at the time of writing).

On this page:

- [Getting native client libraries](#)
 - [Windows](#)
 - [Fedora Linux](#)
 - [Ubuntu and Debian Linux](#)
- [Configuring the native client](#)
 - [Using the FishEye admin console](#)
 - [Using the FishEye config file](#)
- [Performance impact of native client](#)

Related pages:

- [Supported platforms](#)
- [Native support for SVN](#)

Fedora Linux

For Linux systems using the yum package manager (such as Fedora Core 3 and above) you can type the following:

```
yum install subversion-javahl
```

at the Linux command line interface to install the JavaHL bindings for Subversion. Note that this will also install the standard Subversion library, which is required.

Ubuntu and Debian Linux

For Linux systems using the apt-get package manager (such as Debian and Ubuntu) you can type the following:

```
apt-get install libsvn-java
```

at the Linux command line interface to install the JavaHL bindings for Subversion. Note that this will also install the standard Subversion library, which is required.

Configuring the native client

You can configure the path to your native Subversion client either by using the FishEye admin interface, or by editing the `config.xml` configuration file.

The JavaHL bindings include a Java `.jar` file, typically named `javasvnhl.jar`, and a dynamic library such as `libsvnjavah-1.so` or `libsvnjavahl-1.dll`. FishEye must be configured so it can find both the `.jar` and the dynamic library.

If the JavaHL dynamic library is in your library path (such as `%PATH%` on Windows), then FishEye will automatically find it. Otherwise you can tell FishEye where it is, or set the `FISHEYE_LIBRARY_PATH` environm

ent variable before starting FishEye.

Using the FishEye admin console

To set the path to the native client:

1. Go to the [admin area](#) in FishEye.
2. Click **Server** (under 'Global Settings').
3. Click **Edit Details** (under 'Subversion client').
4. Enter the path to the jar file and the dynamic library for the Subversion client:

JAR	The path to the JavaHL .jar.
Dynamic library	The path to the dynamic library, if it is not already on your system's library path.  Due to a bug in earlier versions of the JavaHL bindings, setting this value is ineffective unless you are using a Subversion client 1.2.3 or later.

5. Click **Update**.
6. Restart your FishEye server.

Using the FishEye config file

You can set the path to the native client by editing the <svn-config> section of your FishEye config.xml file . If you change these settings, you need to restart FishEye.

Windows platform example (change path locations as required)

```
<svn-config jar="C:\subversion\lib\svn-javahl.jar"
jnilib="C:\subversion\lib\libsvnjavahl-1.dll"/>
```

Mac OS X platform example (change path locations as required)

```
<svn-config jar="/opt/subversion/lib/svn-javahl/svn-javahl.jar"
jnilib="/opt/subversion/lib/libsvnjavahl-1.dylib"/>
```

Linux platform example (change path locations as required)

```
<svn-config jar="/usr/share/subversion/lib/svn-javahl.jar"
jnilib="/usr/lib/libsvnjavahl-1.so"/>
```

Performance impact of native client

In general, using the JavaHL libraries should improve the speed of FishEye's indexing when compared to SVNKit, primarily because of the advantage of native code. However, some customers have experienced more peak memory usage (some more than 2GB) during indexing when using JavaHL implementations due to greedier processing. This is generally not a major concern, but your own performance requirements may vary.

If you need to allocate more than 2GB for the indexing to complete, please run Fisheye with a 64-bit JDK.

SVNKit client

SVNKit is included in the current FishEye package, and is the default library for interfacing with Subversion. It is generally the easiest to use, and will be used automatically unless another library is specified.

See [Configuring Subversion repositories](#) for more information.

SVNKit supports the 'file://' protocol for FSFS repositories only.

SVNKit sometimes has problems working with Subversion servers which are running older versions, such as 1.1.x. If you see exceptions in FishEye's log file, such as those listed below, you will need to either upgrade your Subversion server to version 1.3 or later, or swap to [use the native client](#).

Example exceptions

- **SEVERE: assert #B**
- **Checksum mismatchwhile reading representation:**

Other Subversion Clients

Native Subversion Clients are also supported by FishEye as an alternative to SVNKit. This process requires additional configuration. Read the [Native Subversion Client instructions](#) for more information.

Re-indexing your repository

Re-indexing involves FishEye performing an intensive scan of the repository contents, so it can more quickly display repository data. Re-indexing may be required in a number of situations, such as after server maintenance, changes in your repository, major setting changes and upgrades to FishEye.

i Due to the complexity of SCM repositories, an indexing scan may take many hours or even days to complete. During this time, your FishEye users may be inconvenienced as they will not be able to view the repository being indexed. Furthermore, Crucible users will not be able to carry out reviews unless you have Crucible set to [store all revisions offline](#).

i How do I mitigate long re-index times when upgrading? [See the FAQ page](#) on this topic.

Performing a manual re-index

Settings and administrative operations for repository indexing are located on the [Indexer](#) page.

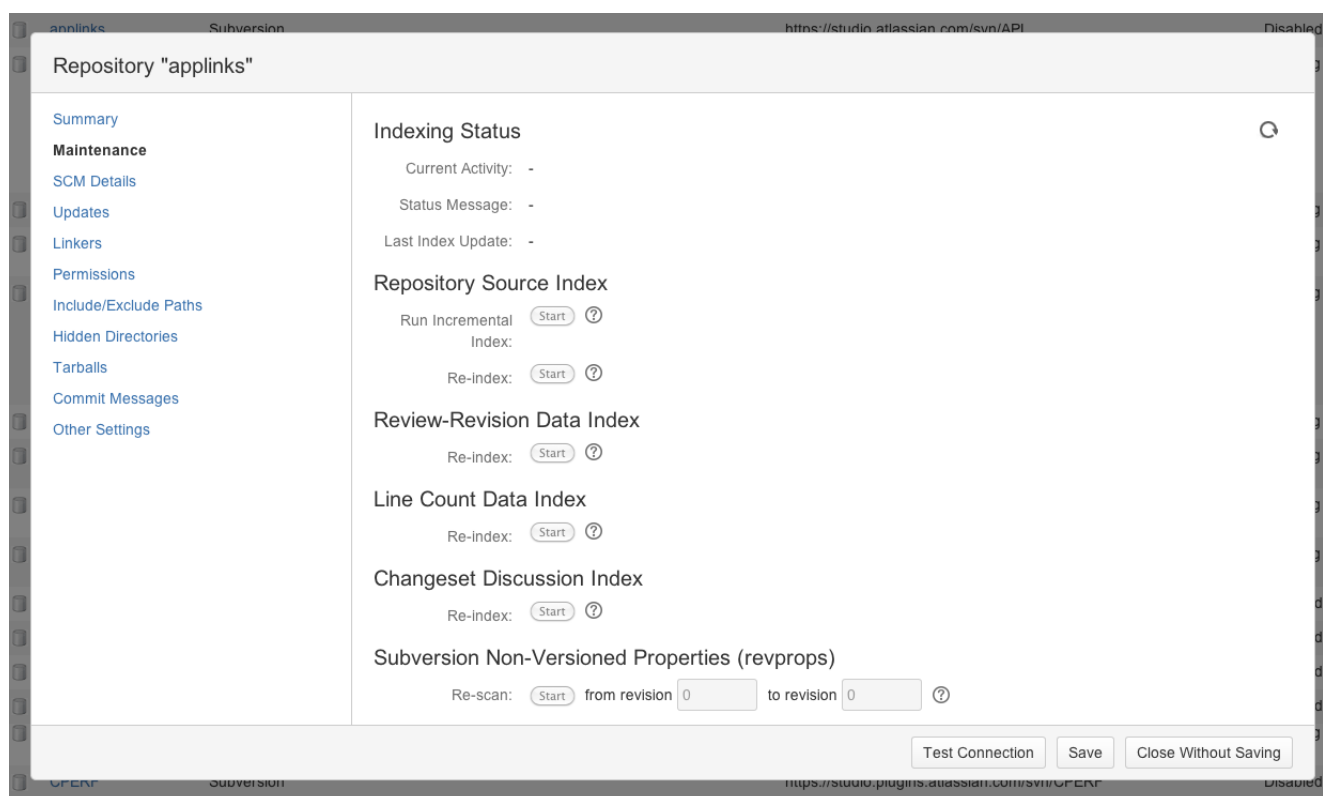
To control indexing for a repository:

1. Click the 'cog' menu in the FishEye header, and choose **Administration** (you'll need to be logged in as an [administrator](#) to see this link).
2. Click the name of the repository.
3. Click **Maintenance** in the left-hand panel.
4. Choose from the following indexing options:

Repository Source Index	
Run Incremental Index	If the repository has already been indexed, this will start an incremental scan, otherwise an "initial" scan. This is especially useful if you have not set the repository to poll automatically, or it is set with a long poll (interval) period.
Re-index	Delete the existing cache and re-index the repository from scratch. For large or slow repositories this may take some time, during which some functionality will be unavailable. This action will also restart the repository.
Re-index Existing Clone	<i>Git and Mercurial repositories</i> - Delete the existing cache and re-index the repository from scratch from the existing clone. For large or slow repositories this may take some time, during which some functionality will be unavailable. This action will also restart the repository.
Re-clone and Re-index	<i>Git and Mercurial repositories</i> - Delete the existing clone and cache and re-index the repository from scratch. Use this if the history of your repository has been rewritten. For large or slow repositories this may take some time, during which some functionality will be unavailable. This action will also restart the repository.
Review-Revision Data Index	<i>Crucible only</i> - Re-index all the Crucible review data associated with the current repository.

Line Count Data Index	Re-index the linecount data used to generate the LOC (Lines Of Code) charts. The linecount data will be recalculated in daily buckets based on the server timezone.
Changeset Discussion Index	Re-index the changeset discussion index, which is used to display changeset discussions in activity streams.
Subversion Non-Version Properties (revprops)	<i>Subversion repositories</i> - Re-scan the non-versioned properties of the repository (e.g. check-in comments if so enabled) and amend the entry in FishEye and Crucible accordingly. Set the revision numbers to scan from and to. In Subversion it is possible to enable non-versioned properties (e.g. the check-in comments) to be updated by committers. When this happens, FishEye will not automatically pick up the updates. By rescanning specific revisions, FishEye will rescan the non-versioned properties and amend the entry in FishEye accordingly.

Screenshot: Indexing options for a Subversion repository



Repository options

You can configure options for repositories that affect settings like caching, permissions, polling period, etc. Some configuration options only apply to individual repositories, whereas others affect all repositories (defaults) in FishEye.

To configure options:

1. Click the 'cog' menu in the FishEye header, and choose **Administration** (you'll need to be logged in as an [administrator](#) to see this link).
2. Under 'Repository Settings', click either:
 - **Defaults** to configure options for all FishEye repositories.
 - **Repositories**, and then the name of the repository, to configure options for a specific FishEye repository.
3. Click a tab to configure those repository options. See the table below for links to further information about particular options.
4. Click **Save**. When configuring options for a specific FishEye repository, you can:
 - Click **Test Connection** to test FishEye's connection to the repository.

- Click **Close Without Saving** to abandon any changes you have made.

	Further information		
Option	Defaults (all repositories)	Specific repositories	Required Action
Summary	NA	Operations	NA
Maintenance	NA	Indexer	NA
SCM details	Checkout URLs	Configuring repository details	Some repository details will prompt you to restart or re-index
Updater	Updater	Updater	Restart
Linkers *	Linkers	Linkers	NA
Permissions	Permissions	Permissions	NA
Include/Exclude paths *	Include Exclude paths	Include Exclude paths	Restart so the changes take effect. Files and directories already indexed will remain visible until a Re-index is performed
Hidden directories *	Hidden directories	Hidden directories	Restart
Tarballs	Tarball settings	Tarball settings	Restart
Commit messages	Commit message syntax	Commit message syntax	NA
Other settings	Watches Properties	Watches Properties	NA
JIRA workflow triggers	JIRA workflow triggers	JIRA workflow triggers	NA

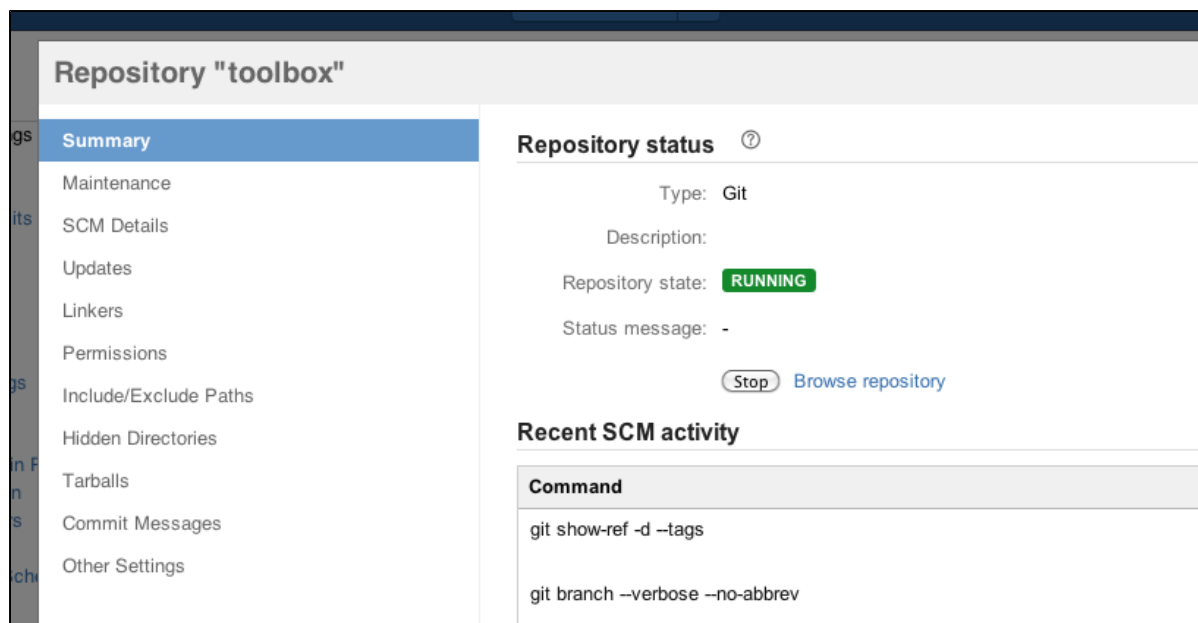
* The default entries will be used in addition to any you define at the repository level.

Some changes will require the repository to be restarted, while others will require the repository to be re-indexed. FishEye will advise you if this is the case when you make the change. You can restart a repository from the 'Repositories' screen, by using the **Restart** function for the desired repository. See [Managing your repositories](#) for further details.

Defaults currently don't prompt for the required action. This is being tracked here:

FE-5633 - Editing Repository Defaults doesn't warn about re-index or restart
[OPEN](#)

Screenshot: Configuring options for the 'toolbox' repository



Authentication

SCMs typically support simple username and password combinations for authentication. However, in FishEye and Crucible, more complex authentication methods are supported for the following repository types:

- Mercurial
- Git

The configuration of these authentication methods in FishEye is described below. Note, the authentication method can also be configured as part of a particular repository's definition.

i Although Subversion supports more complex authentication methods like ssh and ssl certificate authentication, these are not currently configurable in FishEye. When using the bundled svnkit library, these are typically configured using Java property definitions.

On this page:

- [Authentication methods](#)
 - [No authentication](#)
 - [Generate key pair for SSH](#)
 - [Upload private key for SSH](#)
 - [Password](#)
- [Notes](#)
 - [Security considerations when using SSH key pairs](#)
 - [Using ssh-agent](#)
 - [SSH connectivity tools under Windows](#)

Authentication methods

The authentication methods are configured as part of a particular repository definition.

There are four authentication methods for a repository:

- [No authentication](#) — Used where no authentication is required to access a repository.
- [Generate key pair for SSH](#) — A SSH key is generated and managed by FishEye to access the repository.
- [Upload private key for SSH](#) — A SSH key is created and managed externally to FishEye.
- [Password for http\(s\)](#) — This is for the username / password method of authentication with http and https URLs (also known as basic-auth).

When a FishEye repository type supports these authentication methods, you will see an Authentication section during repository configuration (e.g. the "Mercurial Authentication" section in the screenshot below). This section will be available when initially [defining a repository configuration](#) and also in the "SCM Details" section when [maintaining a repository](#).

Screenshot: Selecting an authentication method

New Repository - Page 2 of 3

Mercurial Connection Details

Repository Location: * ?

Mercurial Authentication

Authentication Style:

No authentication

No authentication
Generate key pair for ssh
Upload private key for ssh
Password for http(s)

on to requests. Authentication will be handled transparently by the SCM.

No authentication

This is the simplest authentication method. It means that FishEye can connect to the repository using Git or Mercurial without any credentials or authentication. If you have supplied credentials via another mechanism (such as ssh-agent or a default passphrase-less private key for a ssh connection) Git or Mercurial will use this automatically.

This option is appropriate for,

1. public repositories which grant read-only access to anonymous users, or
2. internal repositories on a secure network which allow anonymous reads to network users, or
3. repositories which are accessible to the FishEye server on the local filesystem.

Screenshot: 'No authentication' selected for a local repository

Mercurial Connection Details

Repository Location: * ?

Mercurial Authentication

Authentication Style:

No authentication

FishEye will not add any authentication to requests. Authentication will be handled transparently by the SCM.

FishEye's access to repositories is always read-only, so it can easily be used with repositories which only require authentication and authorization for write operations.

- If you specify **No authentication** but enter a repository URL which includes a username and password, FishEye will remove the password from the URL and set the authentication method to **Password**, storing the password specified in the password field.

Generate key pair for SSH

If you want to control repository access using SSH keys, you can choose this method of authentication to have FishEye generate and manage the public/private key pair for you. This is the most secure option for ssh access, as the private key is never transmitted across the network.

Click **Generate** to generate the key pair (see "Generating a SSH key pair" screenshot below). FishEye will generate and store the public and private key pair. The key is specific to the repository being indexed.

Screenshot: Generating a SSH key pair

Git Connection Details

Repository Location: * ?

Path: ?

Git Authentication

Authentication Style: Generate key pair for ssh ?

Generate a key pair, then upload the public key to your repository server. Note: SSH keys cannot be used for http(s) urls.

Private Key: Generate

The public key will be displayed to allow you to copy it to your repository server and to associate the key with your user account (see "Public Key of Generated Key displayed" screenshot below). The private key is stored by FishEye and never exposed to users or administrators.

Screenshot: Public key of generated key displayed

Git Authentication

Authentication Style: Generate key pair for ssh ?

Generate a key pair, then upload the public key to your repository server. Note: SSH keys cannot be used for http(s) urls.

Private Key: Private key has been generated by FishEye

Public Key:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQgQCpd/T5PEvaW4q1Nyvob0ZHI4wnv32WIHLU7iwwAZ
sOZumylj+0jM
/yWgECK6d6nDAcdldOoqjjoVgl7q1FojAStuMx+1blzZYY0cY+YAO4gLtz1DPAXHivWLRy5inZT
RjDYmQTgsvhi1d5Jrl3L68d0icsxh/Yk6CoabAYIHvOQ== gittest on FishEye at
http://raleigh.sydney.atlassian.com:8080
```

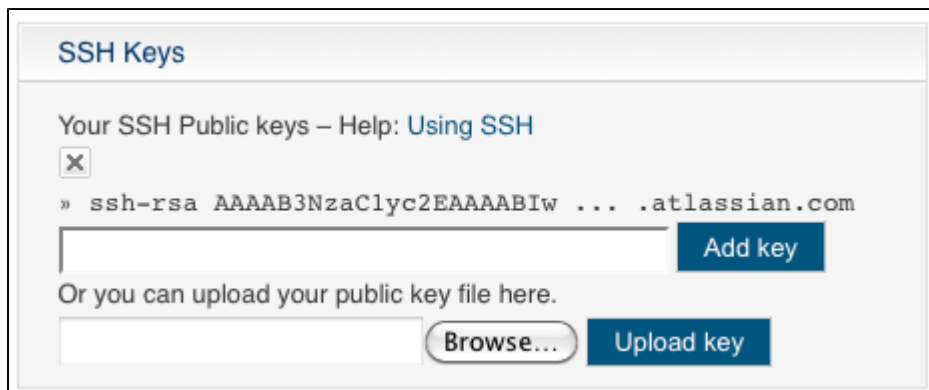
Copy this public key to your repository server
[Remove](#)

If you wish to change the key, click **Remove** and then generate a new key.

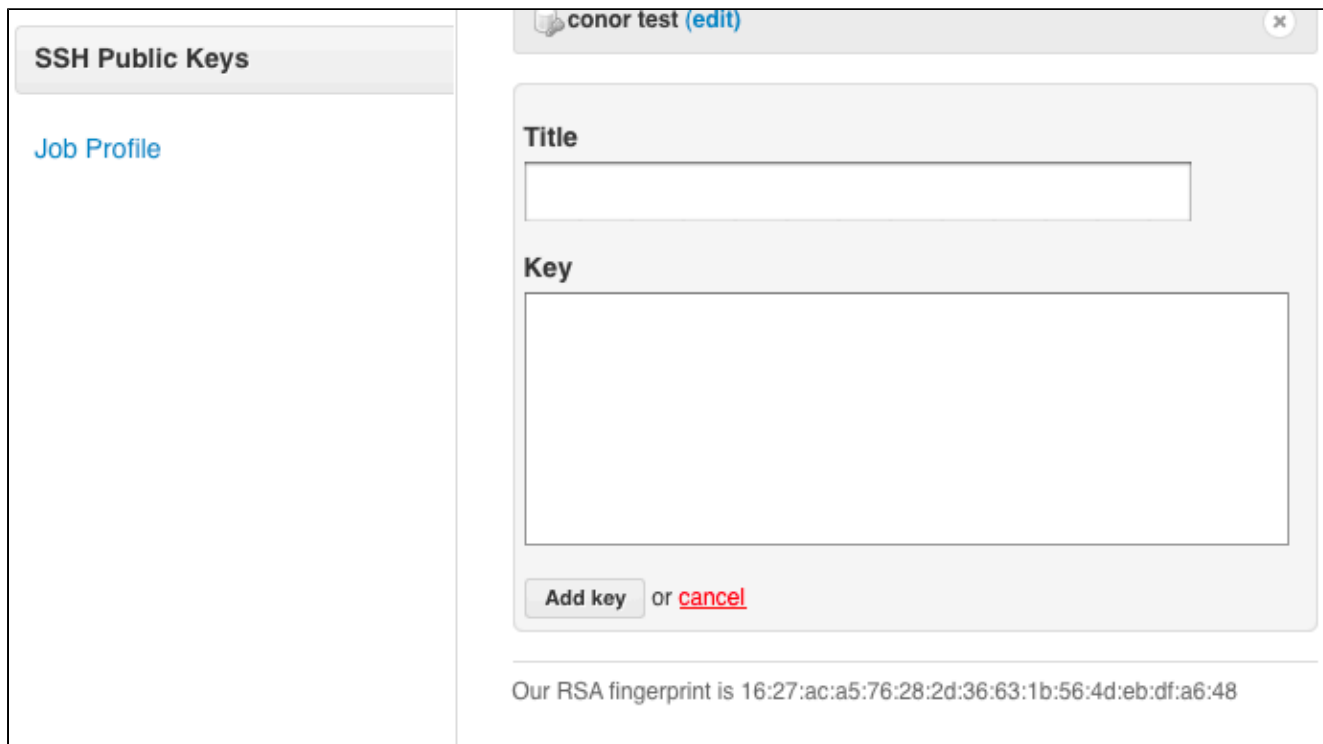
When using SSH keys, you will typically specify a username as part of the URL you use to access the repository.

Public hosting systems such as Bitbucket and GitHub provide simple web-based mechanisms for associating public keys with your account. For these systems, a generic username is used in the repository URL and it is the key that determines the account. See the screenshots below for examples of how to associate keys with Bitbucket and GitHub accounts.

Screenshot: Key management on Bitbucket



Screenshot: Key management on GitHub



Upload private key for SSH

If you are using SSH keys for repository access and already have an SSH key or you would prefer to manage your SSH keys yourself, this authentication method allows you to upload the private key to FishEye. Please note however, that FishEye can only use passphrase-less SSH keys. (To vote for support for private keys with passphrases, see <http://jira.atlassian.com/browse/CRUC-5579>.)

Generating a key within FishEye, as described [above](#), is **our preferred approach** to using SSH keys. We believe it is advisable to use a private key for a single purpose. Different access needs should use different keys. This option should only be used if you must use an existing key.

If you choose to use this option, you will be transmitting your private key across the network to your FishEye server. We **strongly recommend** that you [enable https for Fisheye](#) before you do this.

The private key is uploaded by your web browser file upload operation. As soon as FishEye completes the file upload, it verifies that the provided key is in fact a valid private SSH key and that it does not have a passphrase. To change the stored key, remove the current key by clicking **Remove**, and then upload the new key.

Screenshot: Private key upload

Mercurial Connection Details

Repository Location: * ?

Mercurial Authentication

Authentication Style: ?

Warning: you should understand the risks of uploading private keys. Note: SSH keys cannot be used for http(s) urls.

Private Key: Private key has been uploaded
[Remove](#)

✓ **Tip:** If you are uploading under OSX and using 10.6 or later, you can press command-shift-period to display hidden files and directories. This makes it easier to access ssh key files stored in their default .ssh directory

Password

This authentication method is used when using http/https to access your repository with a username and password. For these repositories, the repository location URL includes the username for the account and the password is configured in the **Password** field (see "Password authentication" screenshot below).

Screenshot: Password authentication

Mercurial Connection Details

Repository Location: * ?

Mercurial Authentication

Authentication Style: ?

Password:

It is possible to specify both the username and password in a repository URL. If you do this FishEye will warn you to remove the password from the URL definition, as it would conflict with the password specified in the password field. When using password authentication, FishEye will always manage the password separately from the repository URL.

Screenshot: Do not provide password in URL

Mercurial Connection Details

Repository Location: * **You have included a password in the repository url which conflicts with the one in the 'password' field. You should remove the password from the url.**
 ?

Mercurial Authentication

Authentication Style: ?

Password:

- For Git repositories, when using http basic authentication, Git must be compiled with libcurl support. If this is not the case, authentication will fail with a message similar to:
error: git was compiled without libcurl support.
You will need to update your Git installation or select an alternative authentication method.
- Password authentication using SSH is not currently supported by FishEye. You must use a key when using SSH.

Notes

Security considerations when using SSH key pairs

The **Generate key pair for SSH** option is the safest option to use, as private keys are never transmitted across the network. If an attacker gains access to your private key, they have access to all the services that are allowed by that key pair.

Since FishEye only requires read access to your accounts, it is best to create a user that only has read access to the repositories you want to index and add the public key to that user's account settings. This way, even if the private key is compromised, while your source can be accessed, you can easily revoke that user's access and/or remove their public keys and the attacker will not be able to change or modify any of your data.

Using ssh-agent

If you are using a Unix-style system with SSH key authentication and do not want to perform key management in FishEye at all, it is possible to launch FishEye in the context of an ssh-agent process. When FishEye launches sub-processes to interact with the repository, the ssh command in those sub-processes will inherit access to the ssh keys in the ssh-agent process. This approach would allow you to use ssh keys with passphrases. Please refer to your operating system documentation for more information on ssh-agent.

SSH connectivity tools under Windows

If your FishEye installation is using Windows, you must use [OpenSSH](#) for SSH connectivity, rather than alternatives like TortoiseHg or Putty.

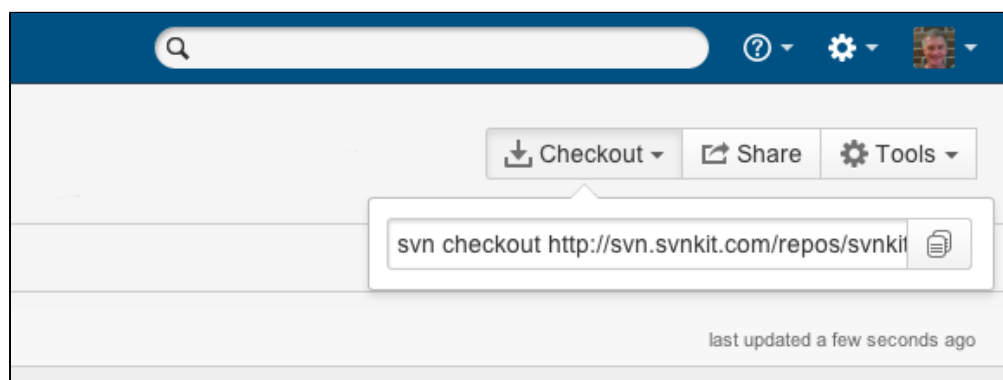
Download and install [Git for Windows](#), as this includes OpenSSH. Be careful about the options you choose during the install:

- When asked how to adjust your PATH environment, select the option 'Run Git and included Unix tool from the Windows Command prompt'. This will ensure that Git and its associated tools (including OPENSSH) are available to the FishEye server via the PATH environment variable.
- When asked for the SSH executable, select the option 'Use OpenSSH'. The FishEye server doesn't work with alternatives like TortoiseHg or Putty.

Checkout URLs

Checkout, or clone, URLs at the top of the page for a FishEye repository allow users to easily clone the source for the repository.

For example, for an SVN repository, a user can click **Checkout (Clone** for a Git repository), and copy the URL:



FishEye admins can configure the display of the checkout URL button, and can control the value of the URL that is displayed to users, both for all repositories in the FishEye instance, and for particular repositories:

- [Configuring the checkout URL for all repositories](#)
- [Configuring the checkout URL for a specific repository](#)

Configuring the checkout URL for all repositories

Repository defaults allow you to control configuration aspects for all repositories in the FishEye instance together.

To control the display of the Checkout/Clone button, go to the admin area and click **Defaults** (under 'Repository Settings').

On the **SCM Details** tab, either select, or clear, the **Show Checkout URL** checkbox:



Repository Defaults

SCM Details Updater Linkers Permissions Include/Exclude Paths Hidden Directories Tarballs Commit Messages Other Settings

Show Checkout URLs

Show users the location where they can check out the source code from a repository. ⓘ

☒ Show Checkout URL

This setting is overridden by the configuration for specific repositories (described below).

Configuring the checkout URL for a specific repository

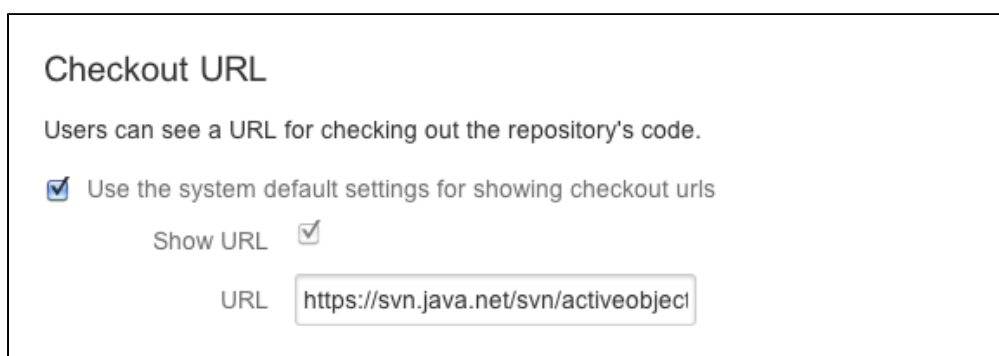
To configure the Checkout/Clone button for a specific repository, go to the admin area and click **Repositories** (under 'Repository Settings').

Click the name of a repository in the list and then **SCM Details** in the left-hand nav panel.

The available fields under 'Checkout URL' at the bottom of the page vary according to the repository type (for example Git, SVN or Perforce), but FishEye will prepopulate fields when it can determine values that are useful.

You may wish to alter the values suggested by FishEye if you want users to access a repository in a different way from that which FishEye uses.

Screenshot below: Checkout URL configuration for an SVN repository



Checkout URL

Users can see a URL for checking out the repository's code.

☒ Use the system default settings for showing checkout urls

Show URL ☒

URL

Screenshot below: Checkout URL configuration for a Perforce repository

Checkout URL

Users can see a URL for checking out the repository's code.

☒ Use the system default settings for showing checkout urls

Show URL ☒

Host

Port

Path

Commit message syntax

FishEye can render Wiki Markup in commit messages. You can enable or disable this either globally, for all repositories, or for individual repositories.

 When using Crucible, you can also have [Wiki Markup rendering in review comments](#) and review descriptions.

On this page:

- [Configure commit message syntax for a specific repository](#)
- [Configure commit message syntax for all repositories](#)

Configure commit message syntax for a specific repository

1. Click the 'cog' menu in the FishEye header, and choose **Administration** (you'll need to be logged in as an [administrator](#) to see this link).
2. Click **Repositories** (in the left-hand panel) and then the name of the repository (in the 'Name' column).
3. Click **Commit Messages** and either use the system defaults or choose markup settings:
 - Commit messages can be displayed as **Plain text** or rendered as **Wiki markup**.
 - Rendering can be limited to commits made after a specified date, if required.
4. Save your changes.

Repository "FE"

Summary
Maintenance
SCM Details
Updates
Linkers
Permissions
Allow (Process)
Hidden Directories
Tarballs
Commit Messages
Other Settings

Commit Messages

FishEye and Crucible can render Wiki markup in commit messages if present.

☐ Use the system default settings for commit message syntax

Interpret commit messages as: Wiki markup

☒ For all commits
☐ Only for commits on or after:
2010-09-24

Test Connection Save and Test Connection Save Close Without Saving

Configure commit message syntax for all repositories

1. In the FishEye admin area, click **Defaults** (in the left-hand panel) and then the **Commit Messages** tab.
2. Choose markup settings:
 - Commit messages can be displayed as **Plain text** or rendered as **Wiki markup**.
 - Rendering can be limited to commits made after a specified date, if required.
3. Save your changes.

Repository Defaults

Linkers Permissions Allow (Process) Hidden Directories Tarballs **Commit Messages** Other Settings

FishEye and Crucible can render Wiki markup in commit messages if present.

Interpret commit messages as: Wiki markup

☒ For all commits
☐ Only for commits on or after:
2010-09-24

Save changes or revert

Configuring repository details

You can define the SCM details for a repository when [adding](#) or [managing](#) a repository. The SCM details will vary depending on the type of repository that you are adding, e.g. the SCM details for a Subversion repository differ from the SCM details for a Git repository.

On this page:

- [Configuring the SCM details for a repository](#)
- [Notes](#)

Related pages:

- [Adding an external repository](#)
- [Repository options](#)

Configuring the SCM details for a repository**To configure the SCM details for a repository:**

1. Click the 'cog' menu in the FishEye header, and choose **Administration** (you'll need to be logged in as an [administrator](#) to see this link).
2. Click **Repositories** (under 'Repository Settings').
3. Click the name of the repository (in the 'Name' column).
4. Click **SCM Details** in the left menu.
5. Update the fields as required:

Description	Enter a description for the repository. This is used to describe the repository in FishEye screens.
Store Diff Info	Select this checkbox, to have FishEye cache information about file diffs in its database. This setting is enabled by default for new repositories. See the Notes section below for more information on this setting.

The rest of the fields will vary depending on the SCM. See the following topics:

- [CVS](#)
- [Git](#)
- [Mercurial](#)
- [Perforce](#)
- [Subversion](#)

Note, the topics above describe adding a repository, however the fields are the same if you are editing an existing repository's settings.

6. Click the **Test Connection** to test your changes.
 - If you are happy with your changes, click **Save**.
 - If you need to abandon your changes, click **Close Without Saving**.

Screenshot: Configuring the SCM details for a Subversion repository

Basic Details

Description:

Store Diff Info: ☒ [?](#)

SVN Connection Details

SVN URL: [?](#)

Path: [?](#)

Username: [?](#)

Password: [?](#)

SVN Options

Block Size: [?](#)

Svn Operation: [?](#)

Timeout:

Throttle Connections/s: [?](#)

Charset: [?](#)

Access Code: [?](#)

Start Revision: [?](#)

Initial Import: [?](#)

Follow Base Moves: ☐ [?](#)

SVN Symbolic Rules

Use Built-in Symbolic Rules: ☐

And then Apply the Following Rules: [Customise](#)

	Regular Expression	Name	Logical Path Prefix	
Trunk				Add
Branches	branches/features/(^)	\${1}-\${2}		Add
Tags				Add

[Test Path](#)

Test Connection

Save

Close Without Saving

Notes

About the 'Store Diff Info' setting

Enabling the **Store Diff Info** setting means that FishEye will store information about file diffs in its database, that is, FishEye will store a summary of what lines are added and removed between subsequent versions of the same file. You will still be able to view file diffs regardless of whether this setting is enabled or disabled.

! Please note, you need to perform a *full re-index* of your repository after enabling this setting for FishEye to collect the diff information for all revisions in your repository.

Please also note that:

- Disabling the **Store Diff Info** setting will *disable* [per-author line graphs](#).
- Disabling the **Store Diff Info** setting also removes the author/revision information in the diff display.
- Disabling the **Store Diff Info** setting will *disable blame calculation when indexing*. See [Blame calculations](#).
- Diff information is always stored for [CVS repositories](#). Note, that a full re-index is required to enable per author charts after upgrading from FishEye 1.4.3 or earlier.
- Enabling the **Store Diff Info** setting will allow FishEye to perform text searches of lines added and removed, in addition to the text search of the trunk head.
- Enabling the **Store Diff Info** setting for [Perforce repositories](#) will force FishEye to make extra requests to

your depot in order to collect the diffs. This may substantially increase the time it takes to scan your repository. If your Perforce repository was created before FishEye 1.5, this setting will be disabled by default.

- Scan times for other repositories, like CVS and Subversion, are not as affected by the **Store Diff Info** setting as Perforce since the diffs are still fetched but disabling this option will still prevent the overhead of indexing the data which can still be significant.

Hidden directories

You can configure unused (deprecated) directories as 'hidden' in FishEye. Hidden directories will not be shown to users in the FishEye user interface, unless a user has specifically elected to view hidden directories in their [user profile](#). This can be useful if you have old directories that you don't want cluttering the screens by default. Please note, FishEye will still index and cache these directories.

 Note, this administration option does not affect the user's option of hiding *empty* directories and deleted files, when browsing a repository. See [Browsing through a Repository](#) for details.

On this page:

- [Configuring hidden directories for a specific repository](#)
- [Configure hidden directories for all repositories](#)
- [Adding a hidden directory](#)

Configuring hidden directories for a specific repository

To configure hidden directories for a repository:

1. Click the 'cog' menu in the FishEye header, and choose **Administration** (you'll need to be logged in as an [administrator](#) to see this link).
2. Click **Repositories**, and then the name of the repository, (in the 'Name' column).
3. Click the **Hidden Directories** tab.
4. Configure the hidden directories for the repository as required:
 - Click **Add** to add a new hidden directory pattern. See the [Adding a Hidden Directory](#) section below for further instructions.
 - Click the cog icon () next to a hidden directory pattern and then **Edit** to edit the pattern.
 - Click the cog icon () next to a hidden directory pattern and then **Delete** to delete the hidden directory pattern.
5. Save your changes.

Configure hidden directories for all repositories

To configure hidden directories for all repositories,

1. Click the 'cog' menu in the FishEye header, and choose **Administration** (you'll need to be logged in as an [administrator](#) to see this link).
2. Click **Defaults**.
3. Click the **Hidden Directories** tab.
4. Configure the hidden directories for all repositories as required:
 - Click **Add** to add a new hidden directory pattern. See the [Adding a Hidden Directory](#) section below for further instructions.
 - Click the cog icon () next to a hidden directory pattern and then **Edit** to edit the pattern.
 - Click the cog icon () next to a hidden directory pattern and then **Delete** to delete the hidden directory pattern.
5. Save your changes.

Screenshot: Configuring hidden directories for all repositories

Repository Defaults

Linkers Permissions Allow (Process) **Hidden Directories** Tarballs Commit Messages Other Settings

Any directory marked as hidden will not be displayed by FishEye and Crucible in its user interface (although it will still be indexed and cached). This is useful to hide unused, deprecated or erroneously committed directories. 🟢

Pattern	Case Sensitive
No hidden directories defined.	
Add...	

Save changes or revert

Adding a hidden directory

To add a new hidden directory,

1. Navigate to the hidden directories for a specific repository or the repository defaults, as described [above](#), and click **Add**.
2. Complete the form as follows:
 - **Pattern** — Enter an Ant Glob pattern for directories that you want to be hidden in FishEye (see [Pattern matching guide](#) for additional information on Ant glob syntax).
 - **Case Sensitive** — By default, hidden directory patterns are case-sensitive but they can be configured to be case-insensitive. If your repository is set to be case-insensitive ([Perforce](#)) then hidden directory patterns will also be case-insensitive, regardless of how the individual patterns are configured.
3. Click **Add**.

Screenshot: Adding a new hidden directory pattern

Add Hidden Directory

Pattern

Case Sensitive: ☐

Add Cancel

Include Exclude paths

By default, FishEye will cache and index your whole repository, and present all of this information to users. You can allow FishEye to process certain parts of your repository and/or disallow it from accessing other parts, by configuring inclusion/exclusion patterns for FishEye to follow. These patterns are set in the 'Include/Exclude Paths' repository option.

On this page:

- [Configuring Include/Exclude paths](#)
 - [Configuring Include/Exclude for a specific repository](#)
 - [Configure Include/Exclude for all repositories](#)
- [Adding an Include](#)
- [Adding an Exclude](#)
- [Notes](#)
 - [Include/Exclude Processing](#)

- [About Setting the Repository Root](#)

Configuring Include/Exclude paths

You can configure the include/exclude paths option for a specific repository or configure linkers for all repositories.

Configuring Include/Exclude for a specific repository

To configure the include/exclude paths option for a repository:

1. Click the 'cog' menu in the FishEye header, and choose **Administration** (you'll need to be logged in as an [administrator](#) to see this link).
2. Click the **Repositories** link. The list of repositories set up in your FishEye instance will be displayed.
3. Click the name of the repository, (under the **Name** column in the list of repositories). The Repository Options for the repository will be displayed in a dialog.
4. Click the **Include/Exclude Paths** tab. The 'Include/Exclude Paths' screen will be displayed (see screenshot below).
5. Configure the includes/excludes for the repository as desired:
 - Click the **Add...** link to add a new include/exclude pattern. See the [Adding an Include](#) and [Adding an Exclude](#) sections below for further instructions.
 - Click the cog icon () next to a linker and click **Edit** from the dropdown menu to edit an include/exclude pattern. The editable 'Linker' dialog is identical to the 'Add a Link' dialog. See the [Adding an Include](#) and [Adding an Exclude](#) sections below for further instructions.
 - Click the cog icon () next to a linker and click **Delete** from the dropdown menu to delete the include/exclude pattern.
6. Save your changes.

Configure Include/Exclude for all repositories

To configure the allow (process) for all repositories:

1. Click the 'cog' menu in the FishEye header, and choose **Administration** (you'll need to be logged in as an [administrator](#) to see this link).
2. Click **Defaults**. The 'Repository Defaults' will be displayed.
3. Click the **Include/Exclude Paths** tab. The 'Include/Exclude Paths' screen will be displayed (see screenshot below).
4. Configure the includes/excludes for all repositories as desired:
 - Click **Add...** to add a new include/exclude pattern. See the [Adding an Include](#) and [Adding an Exclude](#) sections below for further instructions.
 - Click the cog icon () next to a linker and click **Edit** from the dropdown menu to edit an include/exclude pattern. The editable 'Linker' dialog is identical to the 'Add a Link' dialog. See the [Adding an Include](#) and [Adding an Exclude](#) sections below for further instructions.
 - Click the cog icon () next to a linker and click **Delete** from the dropdown menu to delete the include/exclude pattern.
5. Save your changes.

Screenshot: Configuring Include/Exclude Paths for all repositories

Repository Defaults

[Linkers](#)
[Permissions](#)
[Allow \(Process\)](#)
[Hidden Directories](#)
[Tarballs](#)
[Commit Messages](#)
[Other Settings](#)

By default, FishEye and Crucible will cache and index your whole repository, and present all of this information to users. Configure includes and excludes to control which parts of your repository will be accessed. 🌱

Includes

Tree	Case Sensitive
No includes defined. Everything will be included by default.	
Add...	

Excludes

Pattern	Case Sensitive
/CVSROOT/	false 
Add...	

[Save changes](#) or [revert](#)

Adding an Include

The 'Includes' subsection of the 'Include/Exclude Paths' screen defines what subtrees of your repository FishEye will index. FishEye defaults to including 'everything'. If you specify some 'include' directories, then FishEye will process only those directories (and all their subdirectories). For instance, you might want to do this to limit FishEye to the subset of active projects in your repository. Each include specifies the path to a subtree to be processed. Paths are expressed relative to the repository root configured in the repository configuration.

Include paths do not support [Antglobs](#), or wildcards of any type. They are also defined relative to where FishEye connects to your repository. To match the 'tags' subtree, simply use /tags. Please note that [Excludes](#) can still make use of Antglobs.

To add a new include:

1. Navigate to the "Include/Exclude Paths" for a specific repository or the repository defaults, as described [above](#), and click **Add...** to add a new include. The 'Add an Include' dialog will be displayed.
2. Complete the fields on the dialog as follows:
 - **Path Prefix** — Enter the tree of your repository that you want to include. See the [example](#) below.
 - **Case Sensitive** — Tick this checkbox if you want your include to be case-sensitive. By default, Includes are case-sensitive but they can be configured to be case-insensitive. If your repository is set to be case-insensitive ([Perforce](#)) then Includes will also be case-insensitive, regardless of how the individual Includes are configured.

Example:

- Including directories:

```
PROJECT1
```

The code above **includes** /PROJECT1 and all its children (sub-directories and their contents). You could specify /PROJECT1/ and /PROJECT2/ to include both of these directories in FishEye's indexing.

- Including directories with spaces:

```
PROJECT ONE
```

Directories with spaces can be included with a literal space character.

Adding an Exclude

The 'Excludes' subsection of the 'Include/Exclude Paths' screen allows you to specifically exclude files and directories from those which have been included in indexing. FishEye will not process these files and directories. Each exclude is an [Antglob Pattern](#).

To add a new exclude:

1. Navigate to the "Include/Exclude Paths" for a specific repository or the repository defaults, as described [above](#), and click **Add...** to add a new exclude. The 'Add an Exclude' dialog will be displayed.
2. Complete the fields on the dialog as follows:
 - **Pattern** — Enter the pattern for directories/files that you want to exclude. See the [examples](#) below.
 - **Case Sensitive** — Tick this checkbox if you want your exclude to be case-sensitive. By default, Excludes are case-sensitive but they can be configured to be case-insensitive. If your repository is set to be case-insensitive ([Perforce](#)) then Excludes will also be case-insensitive, regardless of how the individual Excludes are configured.

Examples:

- Excluding directories:

```
/PROJECT2/
```

The code above **excludes** /PROJECT2 and all its children (sub-directories and their contents).

```
/PROJECT2014*
```

The code above **excludes** /PROJECT2014 and all its children (sub-directories and their contents), besides other directories containing /PROJECT2014 in their names, e.g. /PROJECT201401, /PROJECT20140205, and so on, having the * working as a wildcard character. It is also possible to use ?, e.g. /PROJECT2014? to exclude paths as /PROJECT20141, /PROJECT20142, with ? replacing one character, with the possibility to have another quantity of ? set, e.g. /PROJECT2014?? to exclude paths as /PROJECT201401, /PROJECT201402, and so on.

- Excluding directories with spaces:

```
/PROJECET?ONE/
```

The code above **excludes** /PROJECT ONE and all its children. The ? wildcard will match any one character, including a space.

- Excluding file types:


```
**/*.*.OBJ
```

The code above **excludes** any OBJ (object) files.

Changes to Includes and Excludes do not take effect until the repository is restarted. If you do not re-index when changing the includes and excluded, files and directories which have been indexed prior to the update will remain visible in FishEye.

Notes

Include/Exclude Processing

When processing includes and excludes, FishEye merges the includes and excludes from the repository itself with those from the repository defaults. The repository's specific includes and excludes take priority over those of the repository defaults. Once merged, FishEye processes include definitions first and then excludes. If there are any includes defined a path must match at least one of those includes to be considered. If there are no include patterns defined, all paths are considered to be included. Once includes have been processed, a path which is a candidate for processing is tested against any defined excludes. If the path matches any of the exclude patterns, the path is excluded and not included in FishEye.

About Setting the Repository Root

When you are setting the Allow (Process), you should be aware that the options on this page only act on the parts of the repository that lie under the level of the repository root, which you configure as a directory location in your repository. In other words, FishEye can only access directories "lower" than the repository root. For example, consider a repository with the following structure:

```
/CORE/2007/LEGACY/  
/CORE/2008/PROJECT1/  
/CORE/2008/PROJECT2/
```

In this case, you could set the repository root (or 'Path') to be /CORE/2008/. In that situation, you would be able to include or exclude the /PROJECT1/ and /PROJECT2/ directories, but the /CORE/2007/LEGACY/ directory would not be available. To have FishEye index all of the directories in this repository, you would need to set the repository root path to be /CORE/. Then, you could use the includes and excludes to add and remove directories under /CORE/ from FishEye's scan. For more information, see the [Subversion configuration page](#) and read the 'Path' options.

Indexer

You can manually perform a number of indexing functions on your repositories in FishEye.

 **Note**, an indexing scan for a repository can take hours or possibly days, depending on the size of the repository, network speed, machine performance and other factors.

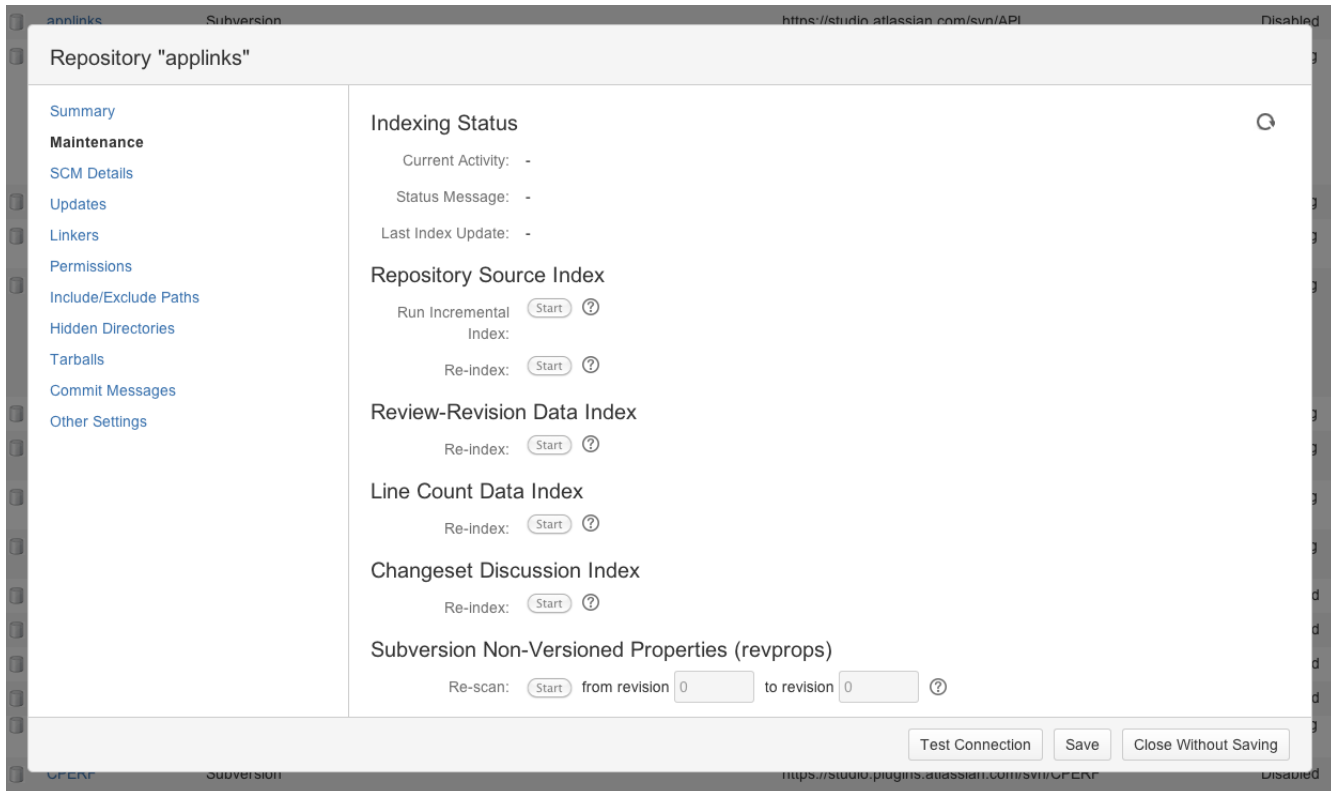
To control indexing for a repository:

1. Click the 'cog' menu in the FishEye header, and choose **Administration** (you'll need to be logged in as an [administrator](#) to see this link).
2. Click the name of the repository.
3. Click **Maintenance** in the left-hand panel.
4. Choose from the following indexing options:

Repository Source Index	

Run Incremental Index	If the repository has already been indexed, this will start an incremental scan, otherwise an "initial" scan. This is especially useful if you have not set the repository to poll automatically, or it is set with a long poll (interval) period.
Re-index	Delete the existing cache and re-index the repository from scratch. For large or slow repositories this may take some time, during which some functionality will be unavailable. This action will also restart the repository.
Re-index Existing Clone	<i>Git and Mercurial repositories</i> - Delete the existing cache and re-index the repository from scratch from the existing clone. For large or slow repositories this may take some time, during which some functionality will be unavailable. This action will also restart the repository.
Re-clone and Re-index	<i>Git and Mercurial repositories</i> - Delete the existing clone and cache and re-index the repository from scratch. Use this if the history of your repository has been rewritten. For large or slow repositories this may take some time, during which some functionality will be unavailable. This action will also restart the repository.
Review-Revision Data Index	<i>Crucible only</i> - Re-index all the Crucible review data associated with the current repository.
Line Count Data Index	Re-index the linecount data used to generate the LOC (Lines Of Code) charts. The linecount data will be recalculated in daily buckets based on the server timezone.
Changeset Discussion Index	Re-index the changeset discussion index, which is used to display changeset discussions in activity streams.
Subversion Non-Version Properties (revprops)	<i>Subversion repositories</i> - Re-scan the non-versioned properties of the repository (e.g. check-in comments if so enabled) and amend the entry in FishEye and Crucible accordingly. Set the revision numbers to scan from and to. In Subversion it is possible to enable non-versioned properties (e.g. the check-in comments) to be updated by committers. When this happens, FishEye will not automatically pick up the updates. By rescanning specific revisions, FishEye will rescan the non-versioned properties and amend the entry in FishEye accordingly.

Screenshot: Indexing options for a Subversion repository



JIRA workflow triggers

FishEye publishes events that can be used to automatically trigger transitions in JIRA Software workflows (when FishEye is [linked](#) with a JIRA Software instance).

- See [Advanced workflow configuration](#) for more information about workflow triggers in JIRA Software.
- See [Transitioning issues in JIRA](#) for other ways to transition JIRA Software issues from FishEye.

FishEye events are published by default. We recommend that you use the latest version of JIRA Software to ensure that duplicate events are handled correctly. JIRA Software automatically removes duplicate commit events (JIRA 6.3.3+) and duplicate branch creation events (JIRA 6.3.11+).

Linkers

FishEye can render any issue IDs or Bug IDs that appear in commit messages or comments as hyperlinks. Users can click the links to quickly navigate to the relevant issue/bug in your issue/bug tracker. This link rendering relies on FishEye detecting certain substrings in commit messages and comments. You can configure the "linker" patterns that FishEye uses to detect these substrings for each repository in FishEye.

Please note, if you have set up [JIRA integration](#) between your FishEye instance and a JIRA server, you will need to disable linkers for that JIRA server. Otherwise, the linkers will override the integration features described on that page (which includes linkers).

On this page:

- [Configuring linkers](#)
 - [Configure linkers for a specific repository](#)
 - [Configure linkers for all repositories](#)
 - [Add a linker](#)
- [Example linkers](#)
 - [JIRA examples](#)
 - [Bamboo examples](#)
 - [Bugzilla examples](#)
- [Configuring the SyntaxDef field](#)

Configuring linkers

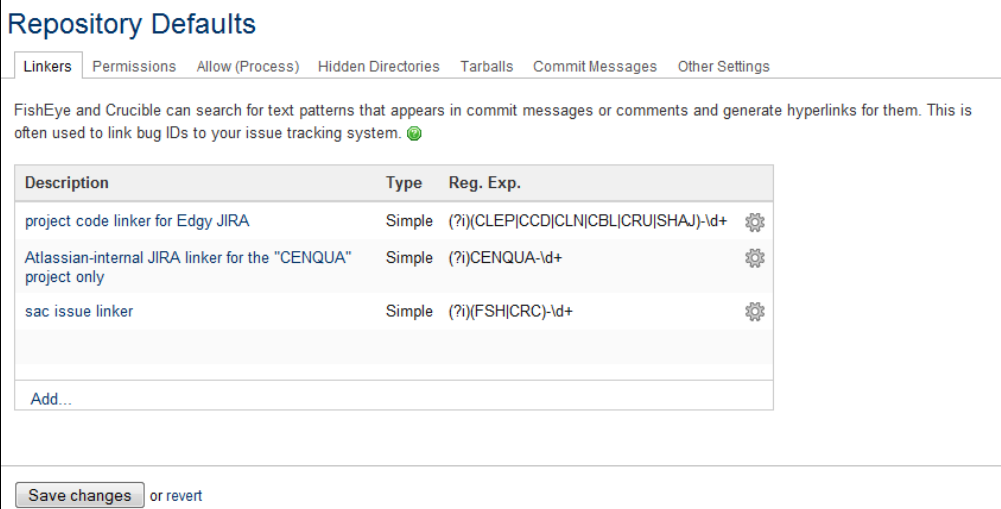
You can configure linkers for a specific repository or configure linkers for all repositories.

Configure linkers for a specific repository

1. Click the 'cog' menu in the FishEye header, and choose **Administration** (you'll need to be logged in as an [administrator](#) to see this link).
2. Click the **Repositories** link. The list of repositories set up in your FishEye instance will be displayed.
3. Click the name of the repository, (under the **Name** column in the list of repositories). The Repository Options for the repository will be displayed in a dialog.
4. Click the **Linkers** tab. The 'Linkers' screen will be displayed (see screenshot below).
5. Configure the linkers for the repository as desired:
 - Click the **Add...** link to add a new linker. See the [Adding a Linker](#) section below for further instructions.
 - Click the cog icon () next to a linker and click **Edit** from the dropdown menu to edit a linker. The editable 'Linker' dialog is identical to the 'Add a Link' dialog, except that you cannot change the link type. See the [Adding a Linker](#) section below for further instructions.
 - Click the cog icon () next to a linker and click **Delete** from the dropdown menu to delete the linker.
6. Save your changes.

Configure linkers for all repositories

1. Click the 'cog' menu in the FishEye header, and choose **Administration** (you'll need to be logged in as an [administrator](#) to see this link).
2. Click **Defaults**. The 'Repository Defaults' will be displayed, with the **Linkers** tab shown (see screenshot below).
3. Configure the linkers for all repositories as required:
 - Click **Add...** to add a new linker. Any new linkers added in the repository defaults will be inherited by all repositories. See the [Adding a Linker](#) section below for further instructions.
 - To edit a linker, click the cog icon () for a linker and click **Edit**. The editable 'Linker' dialog is identical to the 'Add a Link' dialog, except that you cannot change the link type. See the [Adding a Linker](#) section below for further instructions.
 - To delete a linker, click the cog icon () for a linker and choose **Delete**.
4. Save your changes.



Repository Defaults

Linkers Permissions Allow (Process) Hidden Directories Tarballs Commit Messages Other Settings

FishEye and Crucible can search for text patterns that appears in commit messages or comments and generate hyperlinks for them. This is often used to link bug IDs to your issue tracking system. 🐞

Description	Type	Reg. Exp.	
project code linker for Edgy JIRA	Simple	(?i)(CLEPICCD CLN CBL CRU SHAJ)-\d+	
Atlassian-internal JIRA linker for the "CENQUA" project only	Simple	(?i)CENQUA-\d+	
sac issue linker	Simple	(?i)(FSH CRC)-\d+	
Add...			

[Save changes](#) or [revert](#)

Add a linker

1. Navigate to the linkers for a specific repository or the repository defaults, as described [above](#), and click the **Add...** link to add a new linker.

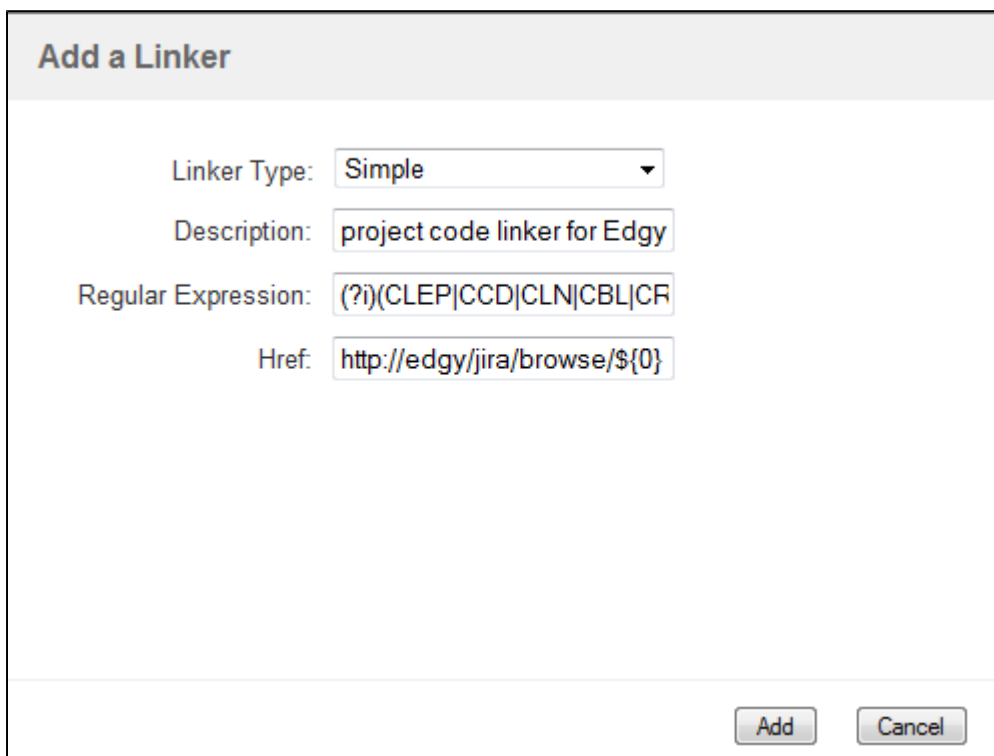
2. Complete the fields on the dialog as follows:

- **Linker Type** — Select 'Simple' or 'Advanced'. Choose 'Simple' unless you are an experienced developer.
- **Description** — Enter a description for your linker.
- **Regular Expression** ('Simple' Linker Type) — Enter the regular expression defining the pattern that FishEye will look for when rendering links. FishEye uses the [Java regular expression language](#), which is based on Perl 5 regular expressions. You can test your regular expressions on this [online test page](#).
✔ Tip: If you want your regex to be case insensitive, put (?i) at the start of the regex.
- **Href** ('Simple' Linker Type) — Enter the destination of the link. For example, `http://jira.mycompany.com/browse/${0}`
- **SyntaxDef** ('Advanced' Linker Type) — See [Configuring the SyntaxDef Field](#) section below.

3. Click **Add**.

4. Save your changes.

✔ **Tip:** See the [Example Linkers](#) section below for examples of linkers to JIRA and Bugzilla servers.



Example linkers

Here are some examples of simple linkers.

JIRA examples

⚠ Do not create linkers for a JIRA server, if you have already set up [JIRA integration](#) between your FishEye instance and that JIRA server.

- To link any occurrence of a JIRA-style issue to JIRA:

```
Regex: [a-zA-Z]{2,}-\d+
Href: http://jirahost:8080/browse/${0}
```

The regular expression above matches any sequence of two or more alphabetical characters, followed by a dash, followed by a number, which comprise the format of JIRA issue IDs (such as AB-123 or ABC-123 or ABCDE-123). Replace `jirahost` with the hostname of the desired JIRA instance.

- To link a specific set of JIRA projects (e.g. JRA, CONF and CRUC) to a JIRA instance:

```
Regex: (JRA|CONF|CRUC)-\d+
Href: http://jirahost:8080/browse/${0}
```

The regular expression above matches only specific JIRA issue keys with any number, like JRA-123 or CONF-123 or CRUC-123. Replace jirahost with the hostname of the desired JIRA instance.

Bamboo examples

- To link to specific Bamboo builds:

```
Regex: (ABC)-[a-zA-Z]+\d+
Href: http://bamboohost/browse/${0}
```

The regular expression above matches Bamboo build IDs like ABC-MAIN-123 or ABC-BRANCH-123. These will then be made links to the build reports in your Bamboo instance. Replace bamboohost with the hostname of the desired bamboo instance.

Bugzilla examples

- To link bug numbers that occur at the start of a line to Bugzilla:

```
Regex: ^BUG: (\d+)
Href: http://bugzilla/bugzilla/show_bug.cgi?id=${1}
```

- To link bug numbers that occur after the word bug and optionally whitespace, ":" or "#" (e.g. Bug123, bug:123, or BUG #123):

```
Regex: (?i)bug[#|\s|:]*(\d+)
Href: http://bugzilla/bugzilla/show_bug.cgi?id=${1}
```

The regular expressions above matches Bugzilla bug IDs. These will then be made links to build reports in your Bamboo instance.

Configuring the SyntaxDef field

 This is an advanced feature, intended for use by experienced developers only.

Please note;

- The **description** is optional.
- You will want to define the three region properties as in the example below.

This example matches numbers that appear after a PR: and might be separated by commas or 'and':

```
PR: 123
PR: 123 456
PR: 123, 456
PR: 123, 456 and 789
```

```
<syntaxdef>
/PR:\s*((\d+\s*(and|,)?\s*)+)/i : {
  nummatcher(${1});
}
context nummatcher {
  /\d+/ : {
    region {
      href="http://issues.apache.org/bugzilla/show_bug.cgi?id=${0}";
    }
  }
}
}
```

Understanding the SyntaxDef example:

- The first regex matches means: match "PR:" followed by a sequence of numbers ("d+") separated by whitespace, 'and' or commas.
- The "context" means in the above match, link each number individually.

Operations

You can carry out operations such as stopping a repository scan, restarting a repository scan, disabling it, etc, via the [repositories list](#) or [repository summary](#) screens.

To access the operations for a repository:

1. Click the 'cog' menu in the FishEye header, and choose **Administration** (you'll need to be logged in as an [administrator](#) to see this link).
 2. Click the **Repositories** link. The list of repositories set up in your FishEye instance will be displayed.
 3. Click the name of the repository, (under the **Name** column in the list of repositories). The Repository Options for the repository will be displayed in a dialog.
 4. A number of repository operations will be available next to the **Actions** field:
 - **Stop** ('Running' repositories only) — Stop the repository.
 - **Restart** ('Running' repositories only) — Restart the repository.
 - **Enable** ('Disabled' repositories only) — Enable the repository.
-  More repository actions are available via the [Repositories list](#). See [Managing your repositories](#).

FishEye reports the current status of the repository as follows:

Status	Description
Running	The repository can be browsed, and is available for indexing of repository changes.
Starting	The start process for the repository has not yet completed.
Stopping	FishEye has received a stop request for the repository, and is waiting for all current operations to complete.
Stopped	The repository can not be browsed, and is not available for indexing, but will start when FishEye next restarts.
Disabled	The repository can not be browsed, and is not available for indexing, and <i>will not be started</i> when FishEye next restarts.

Permissions

You can control access to specific repositories or all repositories in FishEye. Access can be restricted by user groups. You can also allow anonymous access, so that anyone can browse your repositories. For more information, see [Associating a group with a repository](#).

On this page:

- [Configuring permissions for a specific repository](#)
- [Configuring permissions for all repositories](#)

Configuring permissions for a specific repository

You can configure permissions for a specific repository, as described in this section. You can also configure permissions for all repositories at once, [as described below](#).

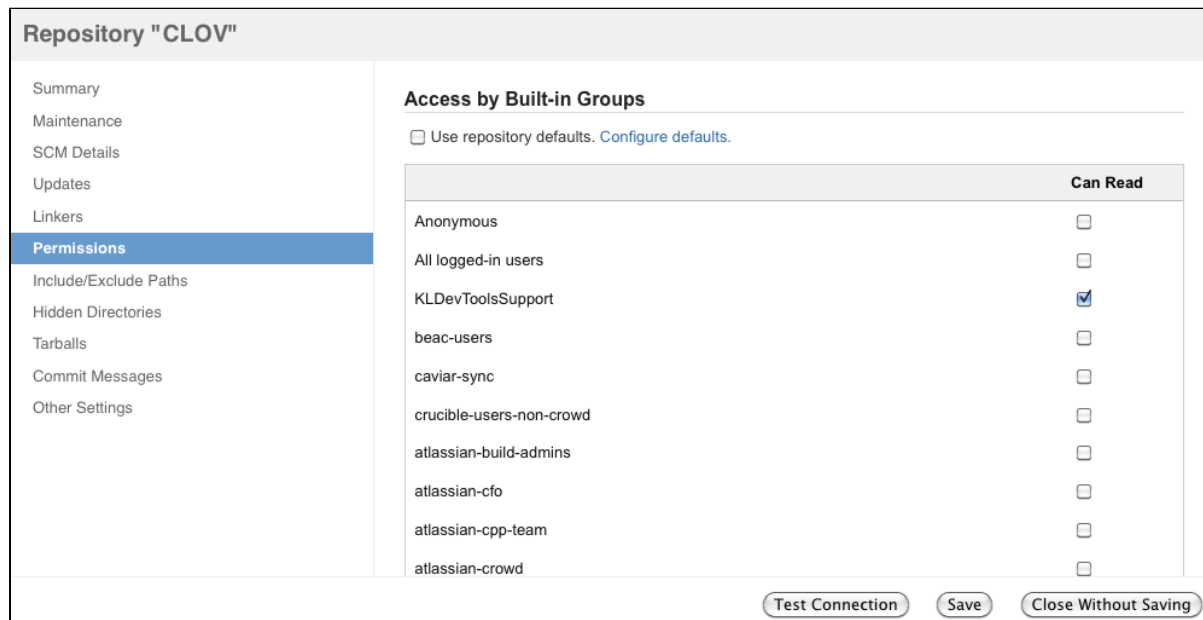
To configure the permissions for a repository:

1. Click the 'cog' menu in the FishEye header, and choose **Administration** (you'll need to be logged in as an [administrator](#) to see this link).
2. Click **Repositories** (under 'Repository Settings').
3. Click the name of the repository (in the 'Name' column).
4. Click **Permissions** in the left-hand panel. See the screenshot below.
5. Configure permissions for the repository as required:

Repository defaults	Select Use repository defaults to have access to this repository controlled by the default permission settings for all repositories .
Anonymous access	Select the Can Read permission for the 'Anonymous' group at the top of the list.
User group access	Select Can Read for the groups to which you wish to grant that permission.

6. Click **Save**.

Screenshot: Configuring permissions for a specific repository



The screenshot shows the 'Repository "CLOV"' configuration page. On the left is a sidebar with a menu containing: Summary, Maintenance, SCM Details, Updates, Linkers, **Permissions** (highlighted), Include/Exclude Paths, Hidden Directories, Tarballs, Commit Messages, and Other Settings. The main content area is titled 'Access by Built-in Groups' and contains a checkbox labeled 'Use repository defaults. [Configure defaults.](#)'. Below this is a table with two columns: a list of user groups and a 'Can Read' column with checkboxes. The groups listed are: Anonymous, All logged-in users, KLDevToolsSupport, beac-users, caviar-sync, crucible-users-non-crowd, atlassian-build-admins, atlassian-cfo, atlassian-cpp-team, and atlassian-crowd. The 'Can Read' checkbox for 'KLDevToolsSupport' is checked. At the bottom right of the main area are three buttons: 'Test Connection', 'Save', and 'Close Without Saving'.

	Can Read
Anonymous	☐
All logged-in users	☐
KLDevToolsSupport	☒
beac-users	☐
caviar-sync	☐
crucible-users-non-crowd	☐
atlassian-build-admins	☐
atlassian-cfo	☐
atlassian-cpp-team	☐
atlassian-crowd	☐

Configuring an LDAP restriction

If the LDAP Authentication type is configured in your instance, you can optionally restrict a repository based on an LDAP query.

When an LDAP Restriction is configured for a repository, then the logic to determine whether you can read from a repository is:

Anon access is allowed:

YES

Logged in AND...

all logged in users are allowed & no LDAP Restriction defined?

YES

authorised by LDAP Restriction?

YES

authorised by group membership?

YES

Otherwise:

NO

See also [LDAP authentication](#).

Configuring permissions for all repositories

You can configure permissions for all repositories at once, as described in this section. You can also configure permissions for a specific repository, as described above.

To configure the permissions for all repositories:

1. Click the 'cog' menu in the FishEye header, and choose **Administration** (you'll need to be logged in as an [administrator](#) to see this link).
2. Click **Defaults** (under 'Repository Settings').
3. Click **Permissions** tab.
4. Configure anonymous access and group permissions for all repositories at once as required:

Anonymous	Check the Can Read permission to allow users without a FishEye account to browse all repositories.
All logged-in users	Check the Can Read permission to allow any user with a FishEye account to browse all repositories.
User groups	Check the Can Read permission for the groups that you wish to be able to browse all repositories.

5. Click **Save changes**.

Repository Defaults

SCM Details
Updater
Linkers
Permissions
Include/Exclude Paths
Hidden Directories
Tarballs
Commit Messages
Other Settings

Access by Built-in Groups

You can instruct FishEye and Crucible to restrict the groups of users who can access the repositories.

	Can Read
Anonymous	☐
All logged-in users	☒
beac-users	☒
caviar-sync	☒
crucible-users-non-crowd	☒

Properties

You can configure a number of default or repository-specific properties for FishEye, including enabling/disabling changeset discussions and enabling/disabling the changelog calendar. A default property is inherited by all repositories. A default property may be overridden at the repository level.

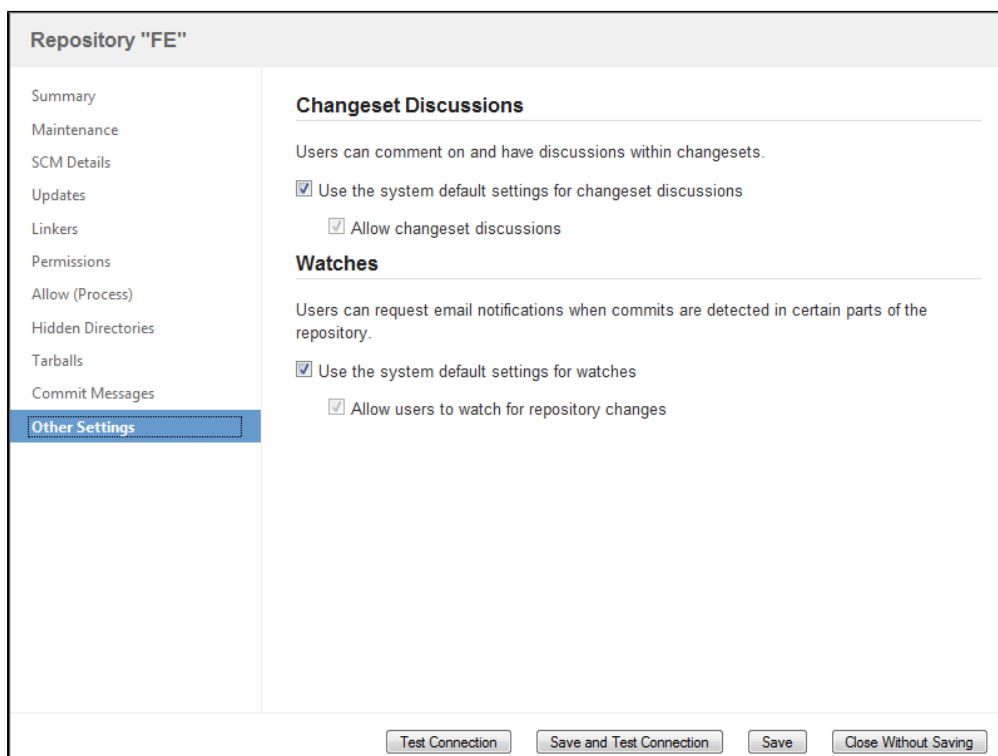
On this page:

- [Configuring properties for a specific repository](#)
- [Configure properties for all repositories](#)

Configuring properties for a specific repository**To configure properties for a repository:**

1. Click the 'cog' menu in the FishEye header, and choose **Administration** (you'll need to be logged in as an [administrator](#) to see this link).
2. Click the **Repositories** link. The list of repositories set up in your FishEye instance will be displayed.
3. Click the name of the repository, (under the **Name** column in the list of repositories). The Repository Options for the repository will be displayed in a dialog.
4. Click the **Other Settings** tab. The other settings screen will be displayed (see screenshot below).
5. Update the settings for this repository as desired:
 - Tick the **Use the system default settings for changeset discussions**, if you want to use the system default settings for changeset discussions, i.e. if you enable/disable changeset discussions in your repository defaults, changeset discussions will be enabled/disabled for this repository respectively.
 - Tick the **Allow changeset discussions** checkbox, if you want to allow users to be able to comment on and have discussions in changesets. This checkbox will only be enabled if the **Use the system default settings for changeset discussions** checkbox is not ticked.
 - Configure settings for watches. See [Watches](#) for more information.
6. Save your changes.

Screenshot: Configuring properties for a specific repository



The screenshot shows the 'Repository "FE"' configuration window. On the left is a sidebar with tabs: Summary, Maintenance, SCM Details, Updates, Links, Permissions, Allow (Process), Hidden Directories, Tarballs, Commit Messages, and **Other Settings** (which is highlighted). The main area is divided into two sections. The top section, 'Changeset Discussions', has a description 'Users can comment on and have discussions within changesets.' and two checkboxes, both of which are checked: 'Use the system default settings for changeset discussions' and 'Allow changeset discussions'. The bottom section, 'Watches', has a description 'Users can request email notifications when commits are detected in certain parts of the repository.' and two checkboxes, both of which are checked: 'Use the system default settings for watches' and 'Allow users to watch for repository changes'. At the bottom of the window are four buttons: 'Test Connection', 'Save and Test Connection', 'Save', and 'Close Without Saving'.

Configure properties for all repositories**To configure properties for all repositories:**

1. Click the 'cog' menu in the FishEye header, and choose **Administration** (you'll need to be logged in as an [administrator](#) to see this link).
2. Click the **Defaults** link. The 'Repository Defaults' will be displayed.

3. Click the **Other Settings** tab. The other settings screen will be displayed (see screenshot below).
4. Update the settings for all repositories as desired:
 - Tick the **Allow changeset discussions** checkbox, if you want to allow users to be able to comment on and have discussions in changesets. This checkbox will only be enabled if the **Use the system default settings for changeset discussions** checkbox is not ticked.
 - Configure settings for watches. See [Watches](#) for more information.
5. Save your changes.

Screenshot: Configuring properties for all repositories

Repository Defaults

Linkers Permissions Allow (Process) Hidden Directories Tarballs Commit Messages **Other Settings**

Changeset Discussions

Users can comment on and have discussions within changesets. 🟢

☒ Allow changeset discussions

Watches

Users can request email notifications when commits are detected in certain parts of the repository. 🟢

☒ Allow users to watch for repository changes

or

Store diff info

The Store Diff Info setting is configured as part of the general repository details. See [Configuring repository details](#) for more information.

About the 'Store Diff Info' Setting

Enabling the **Store Diff Info** setting means that FishEye will store information about file diffs in its database, that is, FishEye will store a summary of what lines are added and removed between subsequent versions of the same file. You will still be able to view file diffs regardless of whether this setting is enabled or disabled.

⚠ Please note, you need to perform a *full re-index* of your repository after enabling this setting for FishEye to collect the diff information for all revisions in your repository.

Please also note that:

- Disabling the **Store Diff Info** setting will *disable* [per-author line graphs](#).
- Disabling the **Store Diff Info** setting also removes the author/revision information in the diff display.
- Disabling the **Store Diff Info** setting will *disable blame calculation when indexing*. See [Blame calculations](#).
- Diff information is always stored for [CVS repositories](#). Note, that a full re-index is required to enable per author charts after upgrading from FishEye 1.4.3 or earlier.
- Enabling the **Store Diff Info** setting will allow FishEye to perform text searches of lines added and removed, in addition to the text search of the trunk head.
- Enabling the **Store Diff Info** setting for [Perforce repositories](#) will force FishEye to make extra requests to your depot in order to collect the diffs. This may substantially increase the time it takes to scan your repository. If your Perforce repository was created before FishEye 1.5, this setting will be disabled by default.
- Scan times for other repositories, like CVS and Subversion, are not as affected by the **Store Diff Info** setting as Perforce since the diffs are still fetched but disabling this option will still prevent the overhead of indexing the data which can still be significant.

Tarball settings

FishEye contains a feature that will build an archive of a directory tree. This feature is disabled by default. The 'Tarball Settings' repository option allows you to customize tarball settings in the Repository Defaults and on a per-repository basis. You can set a limit on the number of files that a tarball can contain, as well as selectively disable the creation of tarballs for certain directories or directory trees.

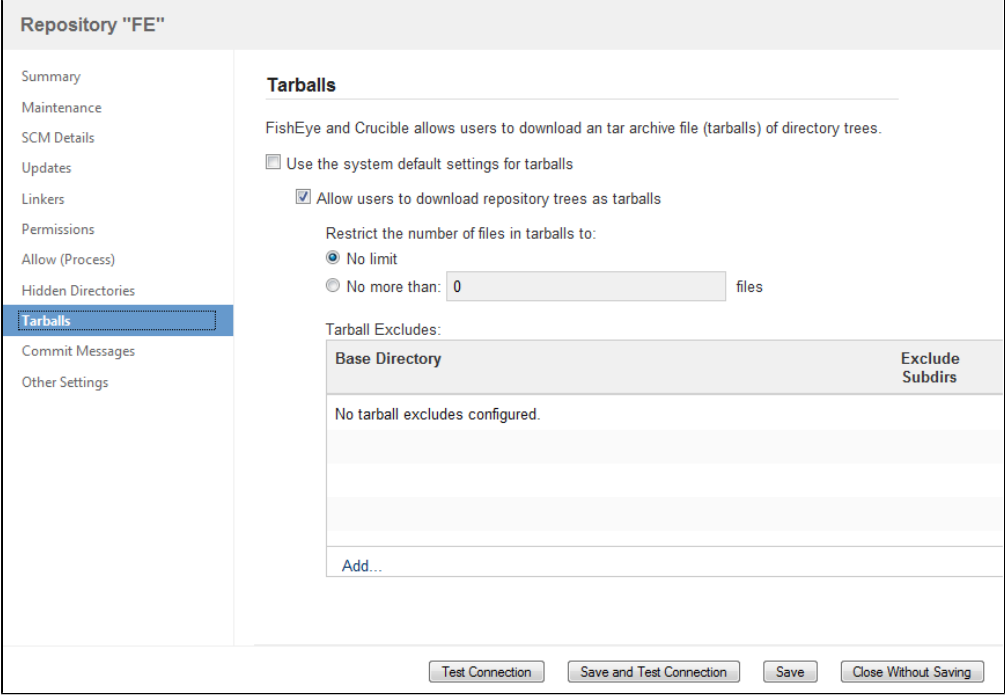
On this page:

- [Configure tarball settings for a specific repository](#)

- [Configure tarball settings for all repositories](#)

Configure tarball settings for a specific repository

1. Click the 'cog' menu in the FishEye header, and choose **Administration** (you'll need to be logged in as an [administrator](#) to see this link).
2. Click **Repositories**, then the name of the repository (in the 'Name' column).
3. Click **Tarballs** in the left-hand panel.
4. Check **Allow users to download repository trees as tarballs**, if you want to allow users to do this. Allow an unlimited number of files, or set a maximum number with **No more than <number> files**.
5. Configure tarball excludes to prevent tarball creation for specified directories or directory trees:
 - To add a new tarball exclude, click **Add**, and then enter a **Base Directory**. Specify **Exclude subdirectories** as required.
 - To edit the exclude pattern, click the cog icon () for a tarball exclude and choose **Edit**. See [Adding a Tarball Exclude](#) below for further instructions.
 - To delete an exclude, click the cog icon () for a tarball exclude and choose **Delete**.
6. Save your changes.



Configure tarball settings for all repositories

1. Click the 'cog' menu in the FishEye header, and choose **Administration** (you'll need to be logged in as an [administrator](#) to see this link).
2. Click **Defaults** in the left-hand panel, then the **Tarballs** tab.
3. Check **Allow users to download repository trees as tarballs**, if you want to allow users to do this. Allow an unlimited number of files, or set a maximum number with **No more than <number> files**.
4. Configure tarball excludes to prevent tarball creation for specified directories or directory trees:
 - To add a new tarball exclude, click **Add**, and then enter a **Base Directory**. Specify **Exclude subdirectories** as required.
 - To edit the exclude pattern, click the cog icon () for a tarball exclude and choose **Edit**. See [Adding a Tarball Exclude](#) below for further instructions.
 - To delete an exclude, click the cog icon () for a tarball exclude and choose **Delete**.
5. Save your changes.

Repository Defaults

[Linkers](#)
[Permissions](#)
[Allow \(Process\)](#)
[Hidden Directories](#)
[Tarballs](#)
[Commit Messages](#)
[Other Settings](#)

FishEye and Crucible allows users to download an tar archive file (tarballs) of directory trees.

☐ Allow users to download repository trees as tarballs

Restrict the number of files in tarballs to:

☒ No limit
 ☐ No more than: files

Tarball Excludes:

Base Directory	Exclude Subdirs
No tarball excludes configured.	
Add...	

[or revert](#)

Text file size limiting

This page describes how text file size limits are applied.

Content indexing

FishEye maintains several indexes which are used to provide fast searching of files in the repository. One of these indexes is used for searching file content. For Subversion repositories only, FishEye places a number of restrictions on this index to keep it manageable and to maintain its usefulness in searching.

- FishEye only indexes text files
- Only the HEAD revision of any file is maintained in the index. New revisions of a file will replace older revisions in the index.
- FishEye only indexes files which are below a size limit
- For most repository types, FishEye only indexes the main branch (i.e. trunk in Subversion, MAIN in CVS, etc)

The limits applied to the content index do not affect the indexing of files in the other indexes maintained by FishEye, notably the metadata index which indexes the file against the commit metadata (comment, author, date, etc). All other FishEye operations are thus unaffected by the application of the content indexing limit.

The default file size limit is 100kB and, for FishEye 2.6 and later versions, this limit was made configurable in the admin user interface. Changing this limit only affects indexing operations. Files which have already been indexed are not affected - i.e. they are not removed from the content index. Only files which are indexed after the change are affected by the changed limit. A re-index is necessary to apply the new limit to all files in the repository.

When a file is not indexed, FishEye will log a warning message in the log file, such as this example from FishEye 3.10:

```
/path/to/file.txt exceeds indexing size limit (300K, limit is 100K).
The contents of this file won't be searchable in FishEye
```

Or the following for older versions:

```
Not indexing contents of HEAD of /path/to/file.txt because its size (6M)
is too large (limit is 100K)
```

Large file viewing

For FishEye 2.7 and later versions, the file size limit for Subversion repositories is also applied to the UI display of text files. Any files which exceed the file size limit have the following additional restrictions:

- The file content is not viewable in the FishEye and Crucible user interfaces.
- The number of lines in the file are not counted towards the Lines of Code (LoC) totals
- Added and Removed text is not indexed and, consequently, is not searchable.

Such files are now marked internally as "too large" within FishEye, and rendered similarly to binary files. A link is available to download the file content and in Crucible file-level review comments can be made.

As above, changing the file size limit does not affect files which have already been indexed. Such files will continue to be treated according to the setting in effect when the file was originally indexed. A re-index is necessary to apply a new limit to files which have already been indexed. This can mean that a large file appears to abruptly acquire the "too large" designation when a new revision is added after upgrading FishEye, or after the file size limit has been changed.

The application of this limit allows FishEye to avoid certain Subversion operations which are resource intensive (typically I/O, memory or time and often all three). Such large text files are typically not practical to view in FishEye and review within Crucible in any case.

Choosing a value for the text file limit

FishEye applied the original 100kB limit as a reasonable compromise for what would typically be the largest text files in a repository. Increasing this value will allow you to index and see the content for larger text files at the cost of increased resource usage.

Updater

You can configure how often FishEye/Crucible checks if there have been new commits to the repository. In most cases, you will not need to change the default settings for this "updater" which polls your repository.

To configure the updater for a repository,

1. Click the 'cog' menu in the FishEye header, and choose **Administration** (you'll need to be logged in as an [administrator](#) to see this link).
2. Click **Repositories**. The list of repositories set up in your FishEye instance will be displayed.
3. Click the name of the repository, (under the 'Name' column in the list of repositories).
4. Click the **Updates** tab.
5. Check **Use the system default settings for updates** to use the default updater settings. In most cases, you can leave this ticked. If you want to customize the updater settings, clear the checkbox and complete the following fields, as required:

Disable polling	If you disable repository polling, FishEye/Crucible will only pick up new commits to this repository if you scan it manually. You can perform a manual scan of the repository via the command line or by using the 'Scan For Updates' function on the repository 'Maintenance' screen.
Polling Interval	Specify how often FishEye should check for new commits to the repository. Use the following terms to specify particular units of time: s, m, h, d, w, mo, y (for seconds, minutes, hours, days, weeks, months and years respectively). For example, 10s. The default value for this field is 60 seconds.

6. The following additional settings can be configured for [CVS](#) repositories. You should not have to change the default values in most cases:

History File	The location of the CVS history file. If you use a relative path, it must be relative to the CVS directory specified for this repository. The default value for this field is <code>./CVSR00T/history</code> .

Full Scan Interval	How often FishEye will perform a full scan of the repository. Use the following terms to specify particular units of time: second, minute, hour, day, week, month, year. For example, 10second. If you do not specify a unit of time, the default unit is days. For example, 15min. Set this field to '0' to disable the periodic full scan (you can still manually perform a full scan via the command line). The default value for this field is 15 minutes.
Strip Prefix	The prefix to strip off files found in the history file. This will make the files relative to this repository's CVS directory. You <i>must</i> configure a strip prefix if the CVS directory specified is not the root of the CVS repository. For example, your CVS is located at /usr/local/cvsroot, but you have specified /usr/local/cvsroot/foo/bar as the CVS directory of this repository. In this case, you will need to set the history file as ../../CVSR00T/history and specify a strip prefix of foo/bar.

FishEye will monitor your CVS history file CVSR00T/history to determine what has changed in your repository. FishEye will also periodically scan the whole repository. CVS is not always configured to create a history file. Talk to your CVS administrator if you need assistance.

- Click **Save**.
- Restart FishEye, if you have changed any values. Your repositories will be scanned in order (depending on the number of threads you have configured) once FishEye starts up. The polling interval and full scan period (CVS only) will then be determined from the time that this initial scan is complete. For example, if you have set your polling interval to one hour, your next scan will begin one hour after your initial scan is complete.

Screenshot: Repository Updates Settings

The screenshot shows the 'Repository "FE"' settings page. On the left is a sidebar with navigation links: Summary, Maintenance, SCM Details, Updates (highlighted), Linkers, Permissions, Allow (Process), Hidden Directories, Tarballs, Commit Messages, and Other Settings. The main content area is titled 'Updates' and contains the following text: 'Settings which affect how often FishEye and Crucible will check if there are new commits to the repository. For most installations there is no need to define a custom updater. ⓘ'. Below this text are two checkboxes: 'Use the system default settings for updates' (checked) and 'Disable polling:' (unchecked). A 'Polling Interval:' field is set to '15s' with a help icon ⓘ. At the bottom of the page are four buttons: 'Test Connection', 'Save and Test Connection', 'Save', and 'Close Without Saving'.

Watches

FishEye has a "watch" notification system that sends email notifications to users when commits are detected. Users can opt in for these notifications by "[watching](#)" a particular repository/activity stream. You can configure FishEye to enable or disable watches for all users for a specific repository or for all repositories.

[Email templates](#) can be customized if required.

On this page:

- [Configure watches for all repositories](#)

Please note, before you can enable watches for your repositories, you must first configure a valid [SMTP server](#).

Please also be aware that no notifications will be sent while the repository is performing the initial indexing. See [Indexer](#).

Configuring watches for a specific repository

To configure watches for a repository:

1. Click the 'cog' menu in the FishEye header, and choose **Administration** (you'll need to be logged in as an [administrator](#) to see this link).
2. Click **Repositories**. The list of repositories set up in your FishEye instance will be displayed.
3. Click the name of the repository, (under the 'Name' column in the list of repositories). The Repository Options for the repository will be displayed in a dialog.
4. Click the **Other Settings** tab. The 'Watches' section will be shown on the screen that is displayed (see screenshot below).
5. Configure the watches for the repository as desired:
 - **Use the system default settings for watches** — Check this if you want to use the system default settings for watches, i.e. if you enable/disable watches in your repository defaults, watches will be enabled/disabled for this repository respectively.
 - **Allow users to watch for repository changes** — Check or clear this to enable/disable watches for the repository. This checkbox will only be enabled if **Use the system default settings for watches** is not checked.
6. Save your changes.

Screenshot: Configuring watches for a specific repository

Repository "FE"

Summary
Maintenance
SCM Details
Updates
Linkers
Permissions
Allow (Process)
Hidden Directories
Tarballs
Commit Messages
Other Settings

Changeset Discussions

Users can comment on and have discussions within changesets.

☒ Use the system default settings for changeset discussions

☒ Allow changeset discussions

Watches

Users can request email notifications when commits are detected in certain parts of the repository.

☒ Use the system default settings for watches

☒ Allow users to watch for repository changes

Test Connection Save and Test Connection Save Close Without Saving

Configure watches for all repositories

To configure watches for all repositories:

1. Click the 'cog' menu in the FishEye header, and choose **Administration** (you'll need to be logged in as an [administrator](#) to see this link).
2. Click the **Defaults** link. The 'Repository Defaults' will be displayed.
3. Click the **Other Settings** tab. The 'Watches' section will be shown on the screen that is displayed (see screenshot below).
4. Configure the watches for all repositories as desired:
 - **Allow users to watch for repository changes** — Check or clear this to enable or disable watches for all repositories that been configured to use the system default settings for watches.
5. Save your changes.

Screenshot: Configuring watches for all repositories

Repository Defaults

[Linkers](#)
[Permissions](#)
[Allow \(Process\)](#)
[Hidden Directories](#)
[Tarballs](#)
[Commit Messages](#)
[Other Settings](#)

Changeset Discussions

Users can comment on and have discussions within changesets. 

☒ Allow changeset discussions

Watches

Users can request email notifications when commits are detected in certain parts of the repository. 

☒ Allow users to watch for repository changes

[Save changes](#) or [revert](#)

Pipelined indexing

Prior to FishEye 3.0, indexing of a repository in FishEye was performed as a serial, monolithic process from the first commit to the latest. The result was that sometimes it could take a long time before the latest (newest) changesets became visible in FishEye. Since the newest changesets are often the most interesting, FishEye could seem to be less than useful until indexing had fully completed.

For FishEye 3.0, and later, a pipelined indexing approach splits the indexing process into separate tasks that can be performed in a phased and concurrent way. This approach allows FishEye to provide core functionality, such as review creation, file browsing, the activity stream and JIRA integration, for all the changesets in the repository, far sooner than in previous versions. You can get on with your work, while FishEye quietly completes the fine details of indexing in the background.

For now, pipelined indexing is only available for Subversion repositories. Other SCMs will be supported in subsequent releases.

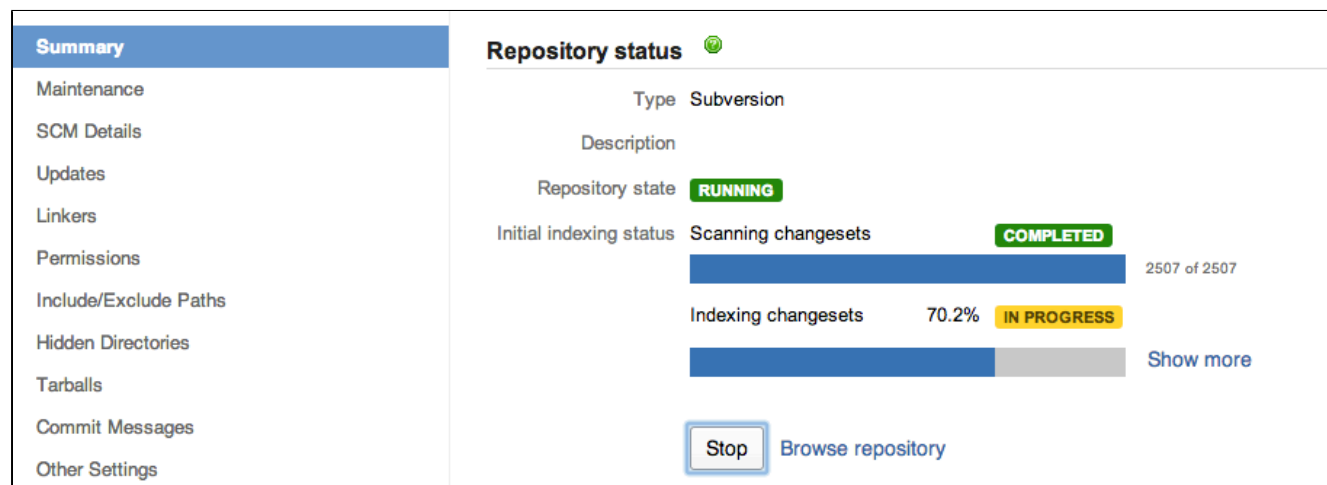
Indexing phases

Pipelined indexing splits the indexing process for a repository into two phases. Tasks within those phases are performed in parallel. When a phase completes, the FishEye functionality that depends on that information then becomes available. The phases, and the functionality associated with them, is shown in the table below:

Phase	What you can do and see
1. Scanning	Create reviews Browse files, revisions and diffs See changeset ancestry Use JIRA integration features View the Activity stream
2. Indexing	Search filenames, content, and diffs See line count data See commit author See file revision ancestry and predecessors Use eyeQL queries

Monitoring progress

When you [add a repository](#) to FishEye, and enable it, pipelined indexing begins immediately. An administrator can monitor indexing progress in the 'Repository status' section of the 'Repository' page:

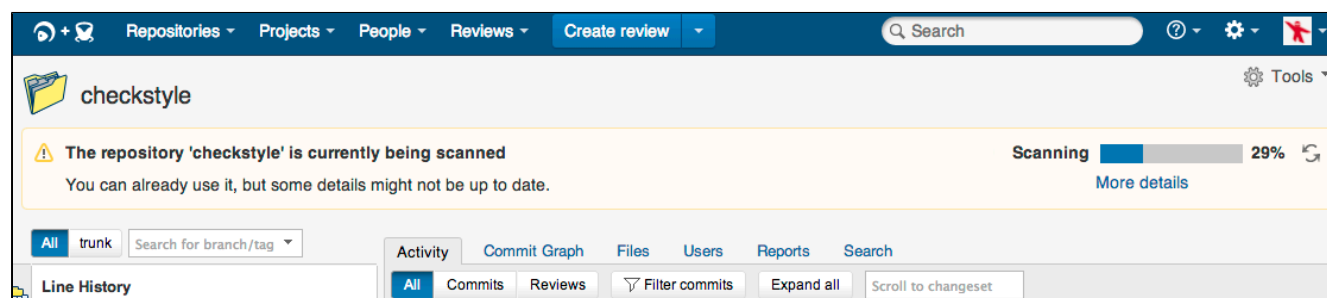


The screenshot shows the 'Repository status' page for a Subversion repository. The repository state is 'RUNNING'. The 'Initial indexing status' shows 'Scanning changesets' as 'COMPLETED' with a progress bar at 2507 of 2507. The 'Indexing changesets' status is 'IN PROGRESS' at 70.2%, with a 'Show more' link. At the bottom, there are 'Stop' and 'Browse repository' buttons.

Click **Show more** to see more details for the indexing process.

You can **Stop**, and **Restart**, indexing if necessary.

When browsing a repository that is currently being indexed, you'll see a progress indicator at the top right of the page:



The screenshot shows the FishEye interface for a repository named 'checkstyle'. A yellow banner at the top states 'The repository 'checkstyle' is currently being scanned' and 'You can already use it, but some details might not be up to date.' A progress bar shows 'Scanning' at 29%. The interface includes navigation tabs like 'All', 'trunk', and 'Search for branch/tag', and a 'Line History' section.

Advanced: configuring pipeline indexing

At start up, FishEye 3.0, and later, auto-configures the pipeline based on the detected environment. This default configuration should suit most users. You may choose to tune aspects of the pipeline operation to suit your environment using the `fisheye.pipeline.name` system properties listed in [JVM system properties](#). To set any of these properties, please see [JVM system properties](#).

In changing these properties, please be aware that there is a single instance of the pipeline, shared between all repositories.

Caution

Changing these values can adversely impact the performance of your FishEye instance. Please proceed with caution.

Managing users and groups in FishEye

You can implement access control using a set of 'built-in' users stored in the FishEye database, or you can have FishEye look in an [external authentication source](#) for users, passwords and permissions.

Note that anonymous access to FishEye is allowed by default. You can disable [anonymous access](#) at a global level and per repository.

To configure external authentication settings, click **Authentication** (under 'Security Settings') in the Admin area.

See [External user directories](#) for more information.

To set up built-in users, choose from the pages below for specific information on settings:

- [Creating a user](#)
- [Creating a group](#)
- [Adding a user to a group](#)
- [Associating a group with a repository](#)
- [Configuring avatar settings](#)
- [Configuring anonymous access](#)
- [Editing a user's details](#)
- [Deleting or deactivating a user](#)
- [Global permissions](#)

FishEye provides a pluggable architecture to allow arbitrary forms of authorization and authentication.

Creating a user

FishEye comes with an internal user directory that is enabled by default at installation.

You can also set up FishEye to [use external user directories](#).

Adding a user is not sufficient to allow them to log in to FishEye. You must explicitly grant them access to FishEye in the [global permissions screen](#).

To add a new user:

1. In the Admin area, click **Users** under 'User Settings'.
2. Click **Add user** at the bottom of the screen.
3. Configure the following settings:

Username	Type the user's login name. You can use the following characters: • letters and numbers • hyphen ('-') • underscore ('_') • 'at' sign ('@')
Display name	The user's display name appears in the user interface.
Email	The email address is where the user will receive notifications .
Password	<i>The user's initial password.</i> The user can easily change their own password later.

4. Click **Add**.

Add New User ?

Information about the new user

Username

Display name

Email

Password

Confirm password

Creating a group

FishEye comes with an internal user directory that is enabled by default at installation.

You can also set up FishEye to [use external user directories](#).

To add a new group:

1. Click **Groups**, under 'Security Settings', in the Admin area.
2. At the bottom of the screen, type the name of your new group into **Group name** and click **Add**.
3. Your new group will now appear in the list of groups.
 To add users to your new group, see [Adding a user to a group](#).

Adding a user to a group

FishEye comes with an internal user directory that is enabled by default at installation.

You can also set up FishEye to [use external user directories](#).

To add a user to a group:

1. In the Admin area, click **Users** under 'User Settings'.
2. Use the filter (type part of their username or email address) to locate the user and then click their username.
3. Click **Edit Groups**, under 'Group Details'. See [Editing a user's details](#).
4. Select group names, and use the **Join** and **Leave** buttons, to add groups to, or remove groups from, the 'Groups' column.
5. Click **Back to username** to save the changes.

Associating a group with a repository

Associating a group with a repository means that only members of that group will be able to access the repository. For more information, please see [Understanding authentication](#).

To associate a group with a repository:

1. Log in to FishEye as an administrator.
2. Click on **Repositories** (under 'Repository Settings') in the left-hand navigation panel.
3. Click on the name of a repository.
4. Click on **Permissions** in the left-hand navigation panel.
5. Select the groups you wish to associate with the repository.
6. Click **Save**.

Repository "CLOV"											
- Summary - Maintenance - SCM Details - Updates - Linkers Permissions - Include/Exclude Paths - Hidden Directories - Tarballs	Access by Built-in Groups ☐ Use repository defaults. Configure defaults. <table> <tr> <th></th><th>Can Read</th></tr> <tr> <td>Anonymous</td><td>☐</td></tr> <tr> <td>All logged-in users</td><td>☐</td></tr> <tr> <td>KLDevToolsSupport</td><td>☒</td></tr> <tr> <td>beac-users</td><td>☐</td></tr> </table>		Can Read	Anonymous	☐	All logged-in users	☐	KLDevToolsSupport	☒	beac-users	☐
	Can Read										
Anonymous	☐										
All logged-in users	☐										
KLDevToolsSupport	☒										
beac-users	☐										

Configuring avatar settings

Every user with a FishEye/Crucible account will have an avatar image displayed next to their name throughout the application. By default, each user will have a plain grey avatar image. Note, you cannot disable avatars.

You can configure whether to use FishEye/Crucible's own avatars or avatars from an external server (e.g. <http://www.gravatar.com>).

In both cases, users can manually upload their own images to use as avatars.

To configure avatar settings:

1. Click the 'cog' menu in the FishEye header, and choose **Administration** (you'll need to be logged in as an [administrator](#) to see this link).

2. Click **Avatars** under the 'User Settings' section.
3. Choose the desired avatar configuration for your FishEye/Crucible instance:

Internal	Use FishEye/Crucible's default avatars, "Charlitors" (Semi-unique avatars resembling charlie) or Joe Doe. We recommend this option if you are running FishEye/Crucible behind a firewall. Users will also be able to upload their own images to use as avatars via their user profile, (see Changing your User Profile). User-defined avatars will override default avatars.
External	Use avatars hosted by an external server (e.g. http://www.gravatar.com). Note that this option prevents users from uploading their own avatar images, via their user profile – see Changing your user profile .
Url	The address of the external avatar server e.g. <code>http://example.com/avatar/</code> Use the same protocol that is used to access FishEye. That is, if FishEye uses the secure HTTP protocol then you need to use 'https' when specifying your external server. If you are using the Gravatar service, we recommend that you use the following URL: <code>https://secure.gravatar.com/avatar/</code>
Suffix	(optional) Specify any parameters supported by the external server. For example, if you wish to control the type of images to be displayed, Gravatar accepts "r=g" ("rating=general"). The suffix must be in the form of a URL query string, starting with an ampersand, e.g. "&r=g"

Note that if you want to use an external avatar server and are using a web server in front of FishEye, you will need to set your [site URL](#) and [proxy host settings](#).

4. Click **Save changes**.

Avatars


☐ Internal
Use FishEye and Crucible's internal avatar server.


☒ External
Choose an avatar server that FishEye and Crucible can use to get custom user avatars.

Url:
 Enter the URL of the external avatar server that uses the Gravatar URL format. E.g. `http://gravatar.com/avatar/`

Suffix:
 Enter any query string suffix supported by the external server. E.g. Gravatar accepts "r=g" ("rating=general").

or [revert](#)

Configuring anonymous access

Anonymous access to FishEye is allowed by default. You can disable anonymous access at the following levels:

- Globally by turning off 'Global Anonymous Access' on the Administration, Authentication page. When this setting is off, an anonymous user just sees a log in page. When it is on they see an activity stream of all repositories which are visible to anonymous users.
- For all repositories which are using the default Security settings (using the [Repository defaults](#)),
- or for particular repositories.

Projects and Reviews

If you are using FishEye with Crucible, anonymous access is also subject to individual project's permissions (see [Creating a permission scheme](#)).

To configure anonymous access globally:

1. Click the 'cog' menu in the FishEye header, and choose **Administration** (you'll need to be logged in as an [administrator](#) to see this link).
2. Click **Defaults** under 'Repository Settings'. The 'Repository Defaults' screen will be displayed.
3. Click the **Permissions** tab.
4. Either check or clear the option *Can Read* in the Anonymous row.

To configure anonymous access per-repository:

See the steps detailed in [Configuring Permissions for a Specific Repository](#).

Editing a user's details

FishEye comes with an internal user directory that is enabled by default at installation.

You can also set up FishEye to [use external user directories](#).

To edit a user's details:

1. In the Admin area, click **Users** under 'User Settings'.
2. Locate the user and click the corresponding **Edit** link.
If the user doesn't initially appear on the screen, type part of their email address and/or select a group to which they belong, and click **Filter**.
3. You can update the following fields:

Display name	To change the user's login name, see Renaming a user .
Email	This is the address to which notifications are sent.

4. Click **Apply**.

Changing a user's password

You can only change the passwords of users in the internal user directory.

To change a user's password:

1. In the Admin area, click **Users** under 'User Settings'.
2. Locate the user and click their username.
If the user doesn't initially appear on the screen, type part of their email address and/or select a group to which they belong, and click **Filter**.
3. Click **Change Password**.
4. Enter, and confirm, the new password.
5. Click **Apply**.

Renaming a user

Renaming a user changes their login name. To change their display name, see [Editing a user's details](#).

Renaming internal users**To rename a user:**

1. In the Admin area, click **Users** under 'User Settings'.
2. Locate the user and click their username.
If the user doesn't initially appear on the screen, type part of their email address and/or select a group to which they belong, and click **Filter**.
3. Click **Rename**.
4. Enter the new username
5. Click **Rename**.

Renaming users when using external directories

When using [external directories](#), you usually rename users in the external directory, not in FishEye itself. FishEye should detect that a user has been renamed during the next directory synchronization, or when the renamed user tries to log in, and associate the modified username with the user data in FishEye, as long as the new username isn't used by any other user in the system.

Deleting or deactivating a user

You can manage which users have access to FishEye/Crucible without deleting them, using the [Global permissions](#) screen.

Deleting the account does not delete their reviews or comments, and can be effectively undone by re-adding them with the same username.

To delete a user's FishEye (and Crucible) account:

1. Choose **Users** from the 'cog' menu.
2. Locate the user in the list and click the corresponding **Delete** link. You can use filters to find a user quickly – type part of their email address or select a group to which they belong, and click **Filter**.

To delete multiple users at once:

1. Choose **Users** from the 'cog' menu.
2. Select check boxes, at the left of the list, for the users you wish to delete.
3. Choose **Delete** from the 'Bulk actions' at the bottom of the list and then click **Execute**.

mac_test_user	In group	Any	Filter
☐ User	Display name	Email	Actions
☐  mac_test_user1	mac_test_user1	mac_test_user1@gmail.com	  ...
☐  mac_test_user2	mac_test_user2	mac_test_user2@gmail.com	  ...
☐  mac_test_user3	mac_test_user3	mac_test_user3@gmail.com	  ...
☐  mac_test_user4	mac_test_user4	mac_test_user4@gmail.com	  ...
☐  mac_test_user5	mac_test_user5	mac_test_user5@gmail.com	  ...
Bulk actions Delete Execute			

Global permissions

For FishEye and Crucible 4.0, and later versions, a user can only log in to FishEye or Crucible if they belong to a group that has access.

You can manage the users who can log in to, and use, FishEye or Crucible using the Global Permissions screen, as [described below](#).

You can see the permissions granted to a particular user on the user's page in the admin area:

User

User Details

Username **joe** **FISHEYE USER** **CRUCIBLE USER**

Display name **joe lowercase**

Email **joe.lowercase@atlassian.com**

Each user with the 'Access to FishEye' permission consumes a FishEye license, each user with the 'Access to Crucible' permission consumes a Crucible license. Users who have neither permission consume no licenses. For instances that combine both FishEye and Crucible, users need to have FishEye access in order to have Crucible access.

Edit account permissions

To edit the account permissions for an existing user group:

1. In the Admin area, click **Global Permissions** under 'Security Settings'.
2. Select, or clear, the permission checkboxes as required.
3. Click in the **Add Groups** field to set permissions for additional groups.

You can remove all permissions for a group by clicking the X at the right-hand end of the row (when you hover there). This will remove that user or group.

Group	Access to Crucible	Access to FishEye
Pick one or more user groups		Fisheye users Add
team-1	☒	☒
team-2	☐	☒
crucible-users	☒	☒
fisheye-users	☐	☒

External user directories

You can connect FishEye to external user directories. This allows you to use existing users and groups stored in an enterprise directory, and to manage those users and groups in one place.

User management functions include:

- **Authentication:** determining which user identity is sending a request to FishEye.
- **Authorization:** determining the access privileges for an authenticated user.
- **User management:** maintaining profile information in user's accounts.
- **Group membership:** storing and retrieving groups, and group membership.

It is important to understand that these are separate components of a user management system. You could use an external directory for any or all of the above tasks.

There are several approaches to consider when using external user directories with FishEye, described briefly below:

- [License considerations](#)
- [LDAP](#)
- [JIRA applications](#)
- [Crowd](#)
- [Multiple directories](#)

- FishEye provides a "read-only" connection to external directories for user management. This means that users and groups, fetched from *any external directory*, can only be modified or updated in the external directory itself, rather than in FishEye.
- Connecting FishEye to your external directory is not sufficient to allow your users to log in to FishEye. You must explicitly grant them access to FishEye in the [global permission screen](#).
- FishEye comes with an internal user directory, already built-in, that is enabled by default at installation.

License considerations

When connecting FishEye to an external directory, be careful not to allow access to FishEye by more users than your FishEye license allows. If the license limit is exceeded, users will not be able to use the application and FishEye will display a warning banner. You can manage which users have access to FishEye, using the [global permission screen](#).

LDAP

You should consider connecting to an LDAP directory server if your users and groups are stored in an

enterprise directory.

There are two common ways of using an external LDAP directory with FishEye:

- For full user and group management, including for user authentication — see [Connecting to an existing LDAP directory](#) for instructions.
- For delegated user authentication only, while using FishEye's internal directory for user and group management — see [Delegating authentication to an LDAP directory](#) for instructions.

FishEye is able to connect to the following LDAP directory servers:

- Microsoft Active Directory
- Apache Directory Server (ApacheDS) 1.0.x and 1.5.x
- Apple Open Directory (Read-Only)
- Fedora Directory Server (Read-Only Posix Schema)
- Novell eDirectory Server
- OpenDS
- OpenLDAP
- OpenLDAP (Read-Only Posix Schema)
- Generic Posix/RFC2307 Directory (Read-Only)
- Sun Directory Server Enterprise Edition (DSEE)
- Any generic LDAP directory server

JIRA applications

You can delegate FishEye user and group management, as well as user authentication, to a [JIRA application](#). This is a good option if you already use a JIRA application in your organization. Note that FishEye can only connect to a JIRA application server running JIRA 4.3 or later.

You should consider using [Atlassian Crowd](#) for more complex configurations with a large number of users.

See [Connecting to JIRA for user management](#) for configuration instructions.

Crowd

You can connect FishEye to [Atlassian Crowd](#) for user and group management, as well as for user authentication.

Crowd is an application security framework that handles authentication and authorization for your web-based applications. With Crowd you can integrate multiple web applications with multiple user directories, with support for single sign-on (SSO) and centralized identity management. See the [Crowd Administration Guide](#).

You should consider connecting to Crowd if you want to use Crowd to manage existing users and groups in multiple directory types, or if you have users of other web-based applications.

See [Connecting to Crowd](#) for configuration instructions.

Multiple directories

When FishEye is connected directly to multiple user directories, where duplicate user names and group names are used across those directories, the effective group memberships that FishEye uses for authorization can be determined using either of these two schemes:

- 'aggregating membership'
- 'non-aggregating membership'.

See [Effective memberships with multiple directories](#) for more information about these two schemes.

Note that:

- Aggregating membership is used by default for new installations of FishEye.
- Authentication, for when FishEye is connected to multiple directories, only depends on the mapped groups in those directories – the aggregation scheme is not involved at all.
- The directory order is significant during the authentication of the user, in cases where the same user exists in multiple directories. When a user attempts to log in, the application will search the directories

in the order specified, and will use the credentials (password) of the *first occurrence of the user* to validate the login attempt.

- For inactive users, FishEye only checks if the user is active in the first (highest priority) directory in which they are found for the purpose of determining authentication. Whether a user is active or inactive does not affect how their memberships are determined.
- When a user is added to a group, they are only added to the first writeable directory available, in priority order.
- When a user is removed from a group, they are only removed from the group in the first directory the user appears in, when non-aggregating membership is used. With aggregating membership, they are removed from the group in *all* directories the user exists in.
- When using Single Sign-On with Crowd and multiple Crowd directories:
 - signed-in users will be validated against the first Crowd directory the user is in
 - users that haven't signed in yet, but have a valid Crowd SSO cookie will be validated against all configured Crowd directories in order

A FishEye admin can change the membership scheme used by FishEye using the following commands:

- To change to *aggregating membership*, substitute your own values for <username>, <password> and <base-url> in this command:

```
curl -H 'Content-type: application/json' -X PUT -d
'{"membershipAggregationEnabled":true}' -u <username>:<password>
<base-url>/rest/crowd/latest/application
```

- To change to *non-aggregating membership*, substitute your own values for <username>, <password> and <base-url> in this command:

```
curl -H 'Content-type: application/json' -X PUT -d
'{"membershipAggregationEnabled":false}' -u
<username>:<password> <base-url>/rest/crowd/latest/application
```

Note that these operations are different from how you make these changes in Crowd. Note also that changing the aggregation scheme can affect the authorization permissions for your users, and how directory update operations are performed.

Connecting to an existing LDAP directory

You can connect FishEye to an existing LDAP user directory, so that your existing users and groups in an enterprise directory can be used in FishEye. The LDAP directory is used for both user authentication and account management.

FishEye is able to connect to the following LDAP directory servers:

- Microsoft Active Directory
- Apache Directory Server (ApacheDS) 1.0.x and 1.5.x
- Apple Open Directory (Read-Only)
- Fedora Directory Server (Read-Only Posix Schema)
- Novell eDirectory Server
- OpenDS
- OpenLDAP
- OpenLDAP (Read-Only Posix Schema)
- Generic Posix/RFC2307 Directory (Read-Only)
- Sun Directory Server Enterprise Edition (DSEE)
- Any generic LDAP directory server

On this page:

- [Synchronization when FishEye is first connected to the LDAP directory](#)
- [Authentication when a user attempts to log in](#)
- [Connecting FishEye](#)
- [Server settings](#)
- [LDAP schema](#)
- [LDAP permission](#)
- [Advanced settings](#)
- [User schema settings](#)
- [Group schema settings](#)
- [Membership schema settings](#)

Connecting FishEye to your external directory is not sufficient to allow your users to log in to FishEye. You must explicitly grant them access to FishEye in the [global permission screen](#).

Synchronization when FishEye is first connected to the LDAP directory

When you first connect FishEye to an existing LDAP directory, the FishEye internal directory is synchronized with the LDAP directory. User information, including groups and group memberships, is copied across to the FishEye directory.

Note that when FishEye is connected to an LDAP directory, you cannot update user details in FishEye. Updates must be done directly on the LDAP directory, perhaps using a LDAP browser tool such as [Apache Directory Studio](#).

Option - Use LDAP filters to restrict the number of users and groups that are synchronized

You can use LDAP filters to restrict the users and groups that are synchronized with the FishEye internal directory. You may wish to do this in order to limit the users or groups that can access FishEye, or if you are concerned that synchronization performance may be poor.

For example, to limit synchronization to just the groups named "fisheye_user" or "red_team", enter the following into the **Group Object Filter** field (see [Group Schema Settings](#) below):

```
(&(objectClass=group)(|(cn=fisheye_user)(cn=red_team)))
```

For further discussion about filters, with examples, please see [How to write LDAP search filters](#). Note that you need to know the names for the various containers, attributes and object classes in your particular directory tree, rather than simply copying these examples. You can discover these container names by using a tool such as [Apache Directory Studio](#).

Authentication when a user attempts to log in

When a user attempts to log in to FishEye, the username and password are passed to the LDAP directory for confirmation. If the password matches that stored for the user, LDAP passes a confirmation back to FishEye, and FishEye logs in the user. During the user's session, all authorizations (i.e. access to FishEye resources such as repositories, reviews and administration screens) are handled by FishEye, based on permissions maintained by FishEye.

Connecting FishEye

To connect FishEye to an LDAP directory:

1. Log in as a user with 'Admin' permission.

2. In the FishEye administration area, click **User Directories** (under 'Accounts').
3. Click **Add Directory** and select either **Microsoft Active Directory** or **LDAP** as the directory type.
4. Configure the directory settings, as described in the tables below.
5. Save the directory settings.
6. Define the directory order by clicking the arrows next to each directory on the 'User Directories' screen.
The directory order has the following effects:
 - The order of the directories is the order in which they will be searched for users and groups.
 - Changes to users and groups will be made only in the first directory where the application has permission to make changes.

Server settings

Setting	Description
Name	Enter a meaningful name to help you identify the LDAP directory server. Examples: • Example Company Staff Directory • Example Company Corporate LDAP
Directory Type	Select the type of LDAP directory that you will connect to. If you are adding a new LDAP connection, the value you select here will determine the default values for many of the options on the rest of screen. Examples: • Microsoft Active Directory • OpenDS • And more.
Hostname	The host name of your directory server. Examples: • ad.example.com • ldap.example.com • opensds.example.com
Port	The port on which your directory server is listening. Examples: • 389 • 10389 • 636 (for example, for SSL)
Use SSL	Check this if the connection to the directory server is an SSL (Secure Sockets Layer) connection. Note that you will need to configure an SSL certificate in order to use this setting.
Username	The distinguished name of the user that the application will use when connecting to the directory server. Examples: • cn=adminstrator,cn=users,dc=ad,dc=example,dc=com • cn=user,dc=domain,dc=name • user@domain.name Ensure that this is an administrator user for the LDAP engine. For example, in Active Directory the user will need to be a member of the built-in Administrators group. The specific privileges for the LDAP user that is used to connect to LDAP are 'bind' and 'read' (user info, group info, group membership, update sequence number, deleted objects). Admin privileges are required because a normal user can't access the uSNChanged attribute and deleted objects container, causing incremental sync to fail silently. This has been reported as CWD-3093.

Password	The password of the user specified above. Note: Connecting to an LDAP server requires that this application log in to the server with the username and password configured here. As a result, this password cannot be one-way hashed - it must be recoverable in the context of this application. The password is currently stored in the database in plain text without obfuscation. To guarantee its security, you need to ensure that other processes do not have OS-level read permissions for this application's database or configuration files.

LDAP schema

Setting	Description
Base DN	The root distinguished name (DN) to use when running queries against the directory server. Examples: - o=example,c=com - cn=users,dc=ad,dc=example,dc=com - For Microsoft Active Directory, specify the base DN in the following format: dc=domain1,dc=local. You will need to replace the domain1 and local for your specific configuration. Microsoft Server provides a tool called ldp.exe which is useful for finding out and configuring the the LDAP structure of your server.
Additional User DN	This value is used in addition to the base DN when searching and loading users. If no value is supplied, the subtree search will start from the base DN. Example: ou=Users
Additional Group DN	This value is used in addition to the base DN when searching and loading groups. If no value is supplied, the subtree search will start from the base DN. Example: ou=Groups

If no value is supplied for **Additional User DN** or **Additional Group DN** this will cause the subtree search to start from the base DN and, in case of huge directory structure, could cause performance issues for login and operations that rely on login to be performed.

LDAP permission

Setting	Description
Read Only	LDAP users, groups and memberships are retrieved from your directory server and can only be modified via your directory server. You cannot modify LDAP users, groups or memberships via the application administration screens.
Read Only, with Local Groups	LDAP users, groups and memberships are retrieved from your directory server and can only be modified via your directory server. You cannot modify LDAP users, groups or memberships via the application administration screens. However, you can add groups to the internal directory and add LDAP users to those groups.

Advanced settings

Setting	Description

Enable Nested Groups	Enable or disable support for nested groups. Some directory servers allow you to define a group as a member of another group. Groups in such a structure are called 'nested groups'. If you are using groups to manage permissions, you can create nested groups to allow inheritance of permissions from one group to its sub-groups.
Manage User Status Locally	If true, you can activate and deactivate users in Crowd independent of their status in the directory server.
Filter out expired users	If true, user accounts marked as expired in ActiveDirectory will be automatically removed. For cached directories, the removal of a user will occur during the first synchronization after the account's expiration date.
Use Paged Results	Enable or disable the use of the LDAP control extension for simple paging of search results. If paging is enabled, the search will retrieve sets of data rather than all of the search results at once. Enter the desired page size – that is, the maximum number of search results to be returned per page when paged results are enabled. The default is 1000 results.
Follow Referrals	Choose whether to allow the directory server to redirect requests to other servers. This option uses the node referral (JNDI lookup <code>java.naming.referral</code>) configuration setting. It is generally needed for Active Directory servers configured without proper DNS, to prevent a 'javax.naming.PartialResultException: Unprocessed Continuation Reference(s)' error.
Naive DN Matching	If your directory server will always return a consistent string representation of a DN, you can enable naive DN matching. Using naive DN matching will result in a significant performance improvement, so we recommend enabling it where possible. This setting determines how your application will compare DN's to determine if they are equal. • If this checkbox is selected, the application will do a direct, case-insensitive, string comparison. This is the default and recommended setting for Active Directory, because Active Directory guarantees the format of DN's. • If this checkbox is not selected, the application will parse the DN and then check the parsed version.
Enable Incremental Synchronization	Enable incremental synchronization if you only want changes since the last synchronization to be queried when synchronizing a directory.  Please be aware that when using this option, the user account configured for synchronization must have read access to: • The <code>uSNC</code> attribute of all users and groups in the directory that need to be synchronized. • The objects and attributes in the Active Directory deleted objects container (see Microsoft's Knowledge Base Article No. 892806 for details). If at least one of these conditions is not met, you may end up with users who are added to (or deleted from) the Active Directory not being respectively added (or deleted) in the application. This setting is only available if the directory type is set to "Microsoft Active Directory".
Synchronization Interval (minutes)	Synchronization is the process by which the application updates its internal store of user data to agree with the data on the directory server. The application will send a request to your directory server every x minutes, where 'x' is the number specified here. The default value is 60 minutes.
Read Timeout (seconds)	The time, in seconds, to wait for a response to be received. If there is no response within the specified time period, the read attempt will be aborted. A value of 0 (zero) means there is no limit. The default value is 120 seconds.

Search Timeout (seconds)	The time, in seconds, to wait for a response from a search operation. A value of 0 (zero) means there is no limit. The default value is 60 seconds.
Connection Timeout (seconds)	This setting affects two actions. The default value is 0. • The time to wait when getting a connection from the connection pool. A value of 0 (zero) means there is no limit, so wait indefinitely. • The time, in seconds, to wait when opening new server connections. A value of 0 (zero) means that the TCP network timeout will be used, which may be several minutes.

User schema settings

Setting	Description
User Object Class	This is the name of the class used for the LDAP user object. Example: • user
User Object Filter	The filter to use when searching user objects. Example: • (&(objectCategory=Person)(sAMAccountName=*)) More examples can be found here and here.
User Name Attribute	The attribute field to use when loading the username. Examples: • cn • sAMAccountName NB: In Active Directory, the 'sAMAccountName' is the 'User Logon Name (pre-Windows 2000)' field. The User Logon Name field is referenced by 'cn'.
User Name RDN Attribute	The RDN (relative distinguished name) to use when loading the username. The DN for each LDAP entry is composed of two parts: the RDN and the location within the LDAP directory where the record resides. The RDN is the portion of your DN that is not related to the directory tree structure. Example: • cn
User First Name Attribute	The attribute field to use when loading the user's first name. Example: • givenName
User Last Name Attribute	The attribute field to use when loading the user's last name. Example: • sn
User Display Name Attribute	The attribute field to use when loading the user's full name. Example: • displayName
User Email Attribute	The attribute field to use when loading the user's email address. Example: • mail
User Password Attribute	The attribute field to use when loading a user's password. Example: • unicodePwd

User Unique ID Attribute	The attribute used as a unique immutable identifier for user objects. This is used to track username changes and is optional. If this attribute is not set (or is set to an invalid value), user renames will not be detected — they will be interpreted as a user deletion then a new user addition. This should normally point to a UUID value. Standards-compliant LDAP servers will implement this as 'entryUUID' according to RFC 4530. This setting exists because it is known under different names on some servers, e.g. 'objectGUID' in Microsoft Active Directory.

Group schema settings

Setting	Description
Group Object Class	This is the name of the class used for the LDAP group object. Examples: - groupOfUniqueNames - group
Group Object Filter	The filter to use when searching group objects. Example: (&(objectClass=group)(cn=*))
Group Name Attribute	The attribute field to use when loading the group's name. Example: cn
Group Description Attribute	The attribute field to use when loading the group's description. Example: description

Membership schema settings

Setting	Description
Group Members Attribute	The attribute field to use when loading the group's members. Example: member
User Membership Attribute	The attribute field to use when loading the user's groups. Example: memberOf
Use the User Membership Attribute, when finding the user's group membership	Check this if your directory server supports the group membership attribute on the user. (By default, this is the 'memberOf' attribute.) - If this checkbox is selected, your application will use the group membership attribute on the user when retrieving the list of groups to which a given user belongs. This will result in a more efficient retrieval. - If this checkbox is not selected, your application will use the members attribute on the group ('member' by default) for the search. - If the Enable Nested Groups checkbox is selected, your application will ignore the Use the User Membership Attribute option and will use the members attribute on the group for the search.

Use the User Membership Attribute, when finding the members of a group

Check this if your directory server supports the user membership attribute on the group. (By default, this is the 'member' attribute.)

- If this checkbox is selected, your application will use the group membership attribute on the user when **retrieving the members of a given group**. This will result in a more efficient search.
- If this checkbox is not selected, your application will use the members attribute on the group ('member' by default) for the search.

Connecting to Crowd

You can configure FishEye to use Atlassian Crowd for user and group management, and for authentication and authorization.

Atlassian Crowd is an application security framework that handles authentication and authorization for your web-based applications. With Crowd you can integrate multiple web applications and user directories, with support for single sign-on (SSO) and centralized identity management. See the [Crowd Administration Guide](#).

Connect to Crowd if you want to use Crowd to manage existing users and groups in multiple directory types, or if you have users of other web-based applications.

Connecting FishEye to your external directory is not sufficient to allow your users to log in to FishEye. You must explicitly grant them access in the [global permission screen](#).

On this page:

- [Server settings](#)
- [Crowd permissions](#)
- [Advanced settings](#)
- [Single sign-on \(SSO\) with Crowd](#)
- [Using multiple directories](#)

To connect FishEye to Crowd:

1. Log in as a user with 'Admin' permission.
2. In the FishEye administration area, click **User Directories** (under 'Security Settings').
3. Click **Add Directory** and select **Atlassian Crowd**.
4. Enter settings, as described below.
5. Test and save the directory settings.
6. Define the directory order, on the **Directories** tab, by clicking the blue up- and down-arrows next to each directory. The directory order has the following effects:
 - The order of the directories is the order in which they will be searched for users and groups.
 - Changes to users and groups will be made only in the first directory where the application has permission to make changes.

Server settings

Setting	Description
Name	A meaningful name that will help you to identify this Crowd server amongst your list of directory servers. Examples: • Crowd Server • Example Company Crowd
Server URL	The web address of your Crowd console server. Examples: • http://www.example.com:8095/crowd/ • http://crowd.example.com

Application Name	The name of your application, as recognized by your Crowd server. Note that you will need to define the application in Crowd too, using the Crowd administration Console. See the Crowd documentation on adding an application .
Application Password	The password which the application will use when it authenticates against the Crowd framework as a client. This must be the same as the password you have registered in Crowd for this application. See the Crowd documentation on adding an application .

Crowd permissions

FishEye offers **Read Only** permissions for Crowd directories. The users, groups and memberships in Crowd directories are retrieved from Crowd and can only be modified from Crowd. You cannot modify Crowd users, groups or memberships using the FishEye administration screens.

Advanced settings

Setting	Description
Enable Nested Groups	Enable or disable support for nested groups. Before enabling nested groups, please check to see if the user directory or directories in Crowd support nested groups. When nested groups are enabled, you can define a group as a member of another group. If you are using groups to manage permissions, you can create nested groups to allow inheritance of permissions from one group to its sub-groups.
Synchronization Interval (minutes)	Synchronization is the process by which the application updates its internal store of user data to agree with the data on the directory server. The application will send a request to your directory server every x minutes, where 'x' is the number specified here. The default value is 60 minutes.

Single sign-on (SSO) with Crowd

Once the Crowd directory has been set up, you can enable or disable Crowd SSO integration by adding the following setting to your \$FISHEYE_INST/config.xml file (you can create missing xml nodes if they don't exist yet). SSO is disabled by default, if not configured in config.xml.

config.xml
<pre> <config> ... <security> ... <ajp... <crowd sso-enabled="true"/> <custom... ... </security> ... </config> </pre>

Note that you will need to correctly set up the domains of the applications involved in SSO. See [Crowd SSO Domain examples](#).

Using multiple directories

When FishEye is connected to Crowd you can map FishEye to multiple user directories in Crowd.

For Crowd 2.8, and later versions, there are two different membership schemes that Crowd can use when multiple directories are mapped to an integrated application, and duplicate user names and group names are used across those directories. The schemes are called 'aggregating membership' and 'non-aggregating membership' and are used to determine the effective group memberships that FishEye uses for *authorization*. See [Effective memberships with multiple directories](#) for more information about these two schemes in Crowd.

Note that:

- *Authentication*, for when FishEye is mapped to multiple directories in Crowd, only depends on the mapped groups in those directories – the aggregation scheme is not involved at all.
- For inactive users, FishEye only checks if the user is active in the first (highest priority) directory in which they are found to determine *authentication*. The membership schemes described above are not used when Crowd determines if a user should have access to FishEye.
- When a user is added to a group, they are only added to the first writeable directory available, in priority order.
- When a user is removed from a group, they are only removed from the group in the first directory the user appears in, when non-aggregating membership is used. With aggregating membership, they are removed from the group in *all* directories the user exists in.

An administrator can set the aggregation scheme that FishEye uses when integrated with Crowd. Go to the **Directories** tab for the FishEye instance in Crowd, and check **Aggregate group memberships across directories** to use the 'aggregating membership' scheme. When the checkbox is clear 'non-aggregating membership' is used.

Note that changing the aggregation scheme can affect the authorization permissions for your FishEye users, and how directory update operations are performed.

Connecting to JIRA for user management

 *This page does not apply to JIRA Cloud; you can't use JIRA Software Cloud to manage your FishEye users.*

You can connect FishEye to an existing Atlassian JIRA instance to delegate FishEye user and group management, and authentication. FishEye provides a "read-only" connection to JIRA for user management. This means that users and groups, fetched from JIRA, can only be modified or updated in that JIRA server, rather than in FishEye.

Choose this option, as an alternative to Atlassian Crowd, for simple configurations with a limited number of users. Note that FishEye can only connect to an instance running JIRA 4.3 or later.

Connecting FishEye and JIRA is a 3-step process:

1. [Set up JIRA to allow connections from FishEye](#)
2. [Set up FishEye to connect to JIRA Software](#)
3. [Set up FishEye users and groups in JIRA Software](#)

Also on this page:

- [Server settings](#)
- [JIRA server permissions](#)
- [Advanced settings](#)

 You need to be an administrator in JIRA and a system administrator in FishEye to perform the following tasks.

1. Setup JIRA to allow connections from FishEye

1. Log in as a user with the 'JIRA Administrators' global permission.
2. For JIRA 4.3.x, select **Other Application** from the 'Users, Groups & Roles' section of the 'Administration' menu.
For later versions, choose **Administration > User management > JIRA User Server**.

3. Click **Add Application**.
4. Enter the **application name** (case-sensitive) and **password** that FishEye will use when accessing JIRA Software.
5. Enter the **IP address** of your FishEye instance. Valid values are:
 - A full IP address, e.g. 192.168.10.12.
 - A wildcard IP range, using CIDR notation, e.g. 192.168.10.1/16. For more information, see the introduction to [CIDR notation on Wikipedia](#) and [RFC 4632](#).
6. Click **Save**.
7. Define the directory order, on the 'User Directories' screen, by clicking the blue up- and down-arrows next to each directory. The directory order has the following effects:
 - The order of the directories is the order in which they will be searched for users and groups.
 - Changes to users and groups will be made only in the first directory where the application has permission to make changes.

2. Setup FishEye to connect to JIRA

1. Log in to FishEye as a user with 'Admin' permission.
2. In the FishEye administration area click **User Directories** (under 'Accounts').
3. Click **Add Directory** and select **Atlassian JIRA**.
4. Enter settings, as described below.
5. Test and save the directory settings.
6. Define the directory order, on the 'User Directories' screen, by clicking the arrows for each directory. The directory order has the following effects:
 - The order of the directories is the order in which they will be searched for users and groups.
 - Changes to users and groups will be made only in the first directory where the application has permission to make changes.

3. Set up FishEye users and groups in JIRA

In order to use FishEye, users must be a member of a group that has the 'Access to FishEye' global permission. Follow these steps to configure your FishEye groups in JIRA:

1. Add a group for FishEye users in JIRA.
2. Add your own username as a member of the above group..
3. Log in to FishEye using and go to the administration area. Click **Global Permissions** (under 'Security Settings'). Assign the appropriate permissions to the JIRA group. See [Global permissions](#).

Server settings

Setting	Description
Name	A meaningful name that will help you to identify this JIRA server in the list of directory servers. Examples: • JIRA Service Desk Server • My Company JIRA
Server URL	The web address of your JIRA server. Examples: • http://www.example.com:8080 • http://jira.example.com
Application Name	The name used by your application when accessing the JIRA server that acts as user manager. Note that you will also need to define your application to that JIRA server, via the ' Other Applications ' option in the 'Users, Groups & Roles' section of the 'Administration' menu.
Application Password	The password used by your application when accessing the JIRA server that acts as user manager.

JIRA server permissions

Setting	Description
Read Only	The users, groups and memberships in this directory are retrieved from the JIRA server that is acting as user manager. They can only be modified via that JIRA server.

Advanced settings

Setting	Description
Enable Nested Groups	Enable or disable support for nested groups. Before enabling nested groups, please check to see if nested groups are enabled on the JIRA server that is acting as user manager. When nested groups are enabled, you can define a group as a member of another group. If you are using groups to manage permissions, you can create nested groups to allow inheritance of permissions from one group to its sub-groups.
Enable Incremental Synchronization	Enable or disable incremental synchronization. Only changes since the last synchronization will be retrieved when synchronizing a directory..
Synchronization Interval (minutes)	Synchronization is the process by which the application updates its internal store of user data to agree with the data on the directory server. The application will send a request to your directory server every x minutes, where 'x' is the number specified here. The default value is 60 minutes.

User management limitations and recommendations

This page describes the optimal configurations and limitations that apply when you are connecting FishEye to JIRA for user management.

On this page:

- [Recommendations for connecting to JIRA for user management](#)
 - [Single Sign-On Across Multiple Applications is Not Supported](#)
 - [Custom Application Connectors are Not Supported](#)
 - [Custom Directories are Not Supported](#)
 - [Load on your JIRA instance](#)
 - [JIRA Cloud applications not supported](#)
 - [Recommendations](#)

Recommendations for connecting to JIRA for user management

Please consider the following limitations and recommendations when connecting to a JIRA server for user management.

Single Sign-On Across Multiple Applications is Not Supported

When you connect to a JIRA application for user management, you will not have single sign-on across the applications connected in this way. JIRA, when acting as a directory manager, does not support SSO.

Custom Application Connectors are Not Supported

JIRA applications, Confluence, FishEye, Crucible and Bamboo can connect to a JIRA server for user management. Custom application connectors will need to use the new REST API.

Custom Directories are Not Supported

Earlier versions of JIRA supported OSUser Providers. It was therefore possible write a special provider to obtain user information from any external user directory. This is no longer the case.

Load on your JIRA instance

If your JIRA instance is already under high load, then using it as a User Server will increase that load.

JIRA Cloud applications not supported

You cannot use JIRA Cloud applications to manage standalone users. Cloud users and users within your self-hosted Atlassian applications need to be managed separately.

Recommendations

Your environment	Recommendation
If all the following are true: • Your JIRA application is not under high load. • You want to share user and group management across just a few applications, such as one JIRA Software server and one Confluence server, or two JIRA servers. • You do not need single sign-on (SSO) between your JIRA application and Confluence, or between two JIRA servers. • You do not have custom application connectors. Or, if you do have them, you are happy to convert them to use the new REST API. • You are happy to shut down all your servers when you need to upgrade your JIRA application.	Your environment meets the optimal requirements for using a JIRA application for user management.
If one or more of the following are true: • If your JIRA application is already under high load. • You want to share user and group management across more than 5 applications. • You need single sign-on (SSO) across multiple applications. • You have custom applications integrated via the Crowd SOAP API, and you cannot convert them to use the new REST API. • You are not happy to shut down all your servers when you need to upgrade JIRA.	We recommend that you install Atlassian Crowd for user management and SSO.
If you are considering creating a custom directory connector to define your own storage for users and groups...	Please see if one of the following solutions will work for you: • If you have written a custom provider to support a specific LDAP schema, please check the supported LDAP schemas to see if you can use one of them instead. • If you have written a custom provider to support nested groups, please consider enabling nested groups in the supported directory connectors instead. • If you have written a custom provider to connect to your own database, please consider loading the data into the application's database instead. • If you need to keep the custom directory connection, please consider whether Atlassian Crowd meets your requirements. See the documentation on Creating a Custom Directory Connector.

Related Topics

Linking FishEye to JIRA

Delegating authentication to an LDAP directory

You can configure FishEye to use an LDAP directory for delegated user authentication while still using FishEye for user and group management.

You can either create new user accounts manually in the LDAP directory, or use the option to automatically create a user account when the user attempts to log in, as described in the [Copy users on login](#) section below.

To connect FishEye to an LDAP directory for delegated authentication:

1. Log in to FishEye as a user with 'Admin' permission.
2. Go to the FishEye administration area and click **User Directories** (under 'Security Settings').
3. Click **Add Directory** and select **Internal with LDAP Authentication** as the directory type.
4. Configure the directory settings, as described in the tables below.
5. Save the directory settings.
6. Define the directory order by clicking the arrows for each directory on the 'User Directories' screen. The directory order has the following effects:
 - The order of the directories is the order in which they will be searched for users and groups.
 - Changes to users and groups will be made only in the first directory where the application has permission to make changes.

Connecting FishEye to your external directory is not sufficient to allow your users to log in to FishEye. You must explicitly grant them access to FishEye in the [global permission screen](#).

On this page:

- [Server settings](#)
- [Manually creating users](#)
- [Copying users on login](#)
- [LDAP schema](#)
- [Advanced settings](#)
- [User schema settings](#)
- [Group schema settings](#)
- [Membership schema settings](#)

Server settings

Setting	Description
Name	A descriptive name that will help you to identify the directory. Examples: • Internal directory with LDAP Authentication • Corporate LDAP for Authentication Only
Directory Type	Select the type of LDAP directory that you will connect to. If you are adding a new LDAP connection, the value you select here will determine the default values for some of the options on the rest of screen. Examples: • Microsoft Active Directory • OpenDS • And more.

Hostname	The host name of your directory server. Examples: • ad.example.com • ldap.example.com • opens.example.com
Port	The port on which your directory server is listening. Examples: • 389 • 10389 • 636 (for example, for SSL)
Use SSL	Check this box if the connection to the directory server is an SSL (Secure Sockets Layer) connection. Note that you will need to configure an SSL certificate in order to use this setting.
Username	The distinguished name of the user that the application will use when connecting to the directory server. Examples: • cn=admin, cn=users, dc=ad, dc=example, dc=com • cn=user, dc=domain, dc=name • user@domain.name
Password	The password of the user specified above.

Manually creating users

Move the delegated authentication directory to the top of the User Directories list and create the user manually (go to **Administration > Users > Create user**). Using this manual method you must currently create a temporary password when creating users.

If you intend to *change* the authentication directory of your users from Internal Directory to Delegated LDAP Authentication you must select the option to "Copy User on Login" since you can't create a new user that has the same username as another user in another directory.

Copying users on login

The settings described in the table below relate to when a user attempts to authenticate with FishEye.

Setting	Description
Copy User on Login	This option affects what will happen when a user attempts to log in. If this box is checked, the user will be created automatically in the internal directory that is using LDAP for authentication when the user first logs in and their details will be synchronized on each subsequent log in. If this box is not checked, the user's login will fail if the user wasn't already manually created in the directory. If you check this box the following additional fields will appear on the screen, which are described in more detail below: • Default Group Memberships • Synchronize Group Memberships • User Schema Settings (described in a separate section below)

Default Group Memberships	This field appears if you check the Copy User on Login box. If you would like users to be automatically added to a group or groups, enter the group name(s) here. To specify more than one group, separate the group names with commas. Each time a user logs in, their group memberships will be checked. If the user does not belong to the specified group(s), their username will be added to the group(s). If a group does not yet exist, it will be added to the internal directory that is using LDAP for authentication. Please note that there is no validation of the group names. If you mis-type the group name, authorization failures will result – users will not be able to access the applications or functionality based on the intended group name. Examples: • confluence-users • bamboo-users,jira-administrators,jira-core-users
Synchronize Group Memberships	This field appears if you select the Copy User on Login checkbox. If this box is checked, group memberships specified on your LDAP server will be synchronized with the internal directory each time the user logs in. If you check this box the following additional fields will appear on the screen, both described in more detail below: • Group Schema Settings (described in a separate section below) • Membership Schema Settings (described in a separate section below)

LDAP schema

Setting	Description
Base DN	The root distinguished name (DN) to use when running queries against the directory server. Examples: • o=example,c=com • cn=users,dc=ad,dc=example,dc=com • For Microsoft Active Directory, specify the base DN in the following format: dc=domain1,dc=local. You will need to replace the domain1 and local for your specific configuration. Microsoft Server provides a tool called ldp.exe which is useful for finding out and configuring the the LDAP structure of your server.
User Name Attribute	The attribute field to use when loading the username. Examples: • cn • SAMAccountName

Advanced settings

Setting	Description
Enable Nested Groups	Enable or disable support for nested groups. Some directory servers allow you to define a group as a member of another group. Groups in such a structure are called 'nested groups'. If you are using groups to manage permissions, you can create nested groups to allow inheritance of permissions from one group to its sub-groups.
Use Paged Results	Enable or disable the use of the LDAP control extension for simple paging of search results. If paging is enabled, the search will retrieve sets of data rather than all of the search results at once. Enter the desired page size – that is, the maximum number of search results to be returned per page when paged results are enabled. The default is 1000 results.

Follow Referrals	Choose whether to allow the directory server to redirect requests to other servers. This option uses the node referral (JNDI lookup <code>java.naming.referral</code>) configuration setting. It is generally needed for Active Directory servers configured without proper DNS, to prevent a 'javax.naming.PartialResultException: Unprocessed Continuation Reference(s)' error.

User schema settings

Note: this section is only visible when **Copy User on Login** is enabled.

Setting	Description
Additional User DN	This value is used in addition to the base DN when searching and loading users. If no value is supplied, the subtree search will start from the base DN. Example: ou=Users
User Object Class	This is the name of the class used for the LDAP user object. Example: user
User Object Filter	The filter to use when searching user objects. Example: (&(objectCategory=Person)(sAMAccountName=*))
User Name RDN Attribute	The RDN (relative distinguished name) to use when loading the username. The DN for each LDAP entry is composed of two parts: the RDN and the location within the LDAP directory where the record resides. The RDN is the portion of your DN that is not related to the directory tree structure. Example: cn
User First Name Attribute	The attribute field to use when loading the user's first name. Example: givenName
User Last Name Attribute	The attribute field to use when loading the user's last name. Example: sn
User Display Name Attribute	The attribute field to use when loading the user's full name. Example: displayName
User Email Attribute	The attribute field to use when loading the user's email address. Example: mail

Group schema settings

Note: this section is only visible when both **Copy User on Login** and **Synchronize Group Memberships** are enabled.

Setting	Description
Additional Group DN	This value is used in addition to the base DN when searching and loading groups. If no value is supplied, the subtree search will start from the base DN. Example: ou=Groups

Group Object Class	This is the name of the class used for the LDAP group object. Examples: • groupOfUniqueNames • group
Group Object Filter	The filter to use when searching group objects. Example: • (objectCategory=Group)
Group Name Attribute	The attribute field to use when loading the group's name. Example: • cn
Group Description Attribute	The attribute field to use when loading the group's description. Example: • description

Membership schema settings

Note: this section is only visible when both **Copy User on Login** and **Synchronize Group Memberships** are enabled.

Setting	Description
Group Members Attribute	The attribute field to use when loading the group's members. Example: • member
User Membership Attribute	The attribute field to use when loading the user's groups. Example: • memberOf
Use the User Membership Attribute, when finding the user's group membership	Check this box if your directory server supports the group membership attribute on the user. (By default, this is the 'memberOf' attribute.) • If this box is checked, your application will use the group membership attribute on the user when retrieving the members of a given group. This will result in a more efficient retrieval. • If this box is not checked, your application will use the members attribute on the group ('member' by default) for the search.

Migrating to an external database

This page provides an overview of connecting FishEye to an external database. Atlassian recommends that you use an external database for production installations, for the following reasons:

- **Improved protection against data loss:** The FishEye built-in database, which runs [HSQLDB](#), is susceptible to data loss during system crashes. External databases are generally more resistant to data loss during a system crash. HSQLDB is not supported in production environments and should only be used for evaluation purposes.
- **Performance & scalability:** If you have a large number of users on your FishEye instance, running the database on the same server as FishEye may slow it down. When using the embedded database, the database will always be hosted and run on the same server as FishEye.
- **Data stored in the FishEye database:** The FishEye database stores information besides the cache for repository scans. Specifically, user data and user preferences information.

Note that when they are used together, FishEye and Crucible share the same external database.

On this page:

- [Supported databases](#)
- [Support for other databases](#)
- [Notes](#)

Related pages:

- [Configuring the database connection pool](#)

Supported databases

You can use a number of alternatives to the built-in HSQLDB database for storing FishEye and Crucible's relational data. The supported databases versions are listed on the [Supported platforms](#) page.

The pages linked below detail the steps required to migrate to an external database:

- [Migrating to MySQL](#)
- [Migrating to Oracle](#)
- [Migrating to PostgreSQL](#)
- [Migrating to SQL Server](#)
- [Configuring the database connection pool](#)

Support for other databases

If you are using another database product that you would like to see supported, please create a [JIRA issue](#) for it under the Crucible project.

Notes

FishEye uses Read Committed transaction isolation. There is no requirement to configure this explicitly when setting up an external database - FishEye will configure the transaction isolation when connecting to the database.

Migrating to MySQL

This page describes how to use FishEye/Crucible with both MySQL Enterprise Server and MySQL Community Server. Note that when they are used together, FishEye and Crucible share the same external database.

To switch to a MySQL database, install MySQL and then follow the steps below. Please note that during the migration of database servers, the FishEye/Crucible instance will not be available to users or to external API clients.

 Note that for FishEye 2.9+, the JDBC driver for MySQL is *not bundled* with FishEye/Crucible (due to [licensing restrictions](#)).

MySQL 5.6.x and 5.7.x compatibility

For MySQL 5.6, FishEye/Crucible requires at least version 5.6.11.

For MySQL 5.7, FishEye/Crucible requires at least version 5.7.5.

Prerequisites

To start with:

1. Install a *supported version* of MySQL. Check the [Supported platforms](#) page for the exact versions that are supported. Note that MariaDB and Percona variants of MySQL are not supported, and are known to cause issues when used with FishEye.
2. Download and install the JDBC driver, if necessary. Note that for FishEye 2.9+, the JDBC driver for MySQL is *not bundled* with FishEye/Crucible (due to [licensing restrictions](#)).
 - a. Download the **MySQL Connector/J JDBC driver** from the [MySQL download website](#).
 - b. Expand the downloaded **zip/tar.gz file**.
 - c. Copy the `mysql-connector-java-x.y.zz-bin.jar` file to your `FISHEYE_INST/lib` directory. If the `lib/` directory doesn't already exist, you will need to create it.

d. Restart FishEye/Crucible.

On this page:

- [Prerequisites](#)
- [Step 1. Create a MySQL database](#)
- [Step 2. Configure FishEye/Crucible to use MySQL, and migrate data](#)

Related pages:

- [Migrating to PostgreSQL](#)
- [Migrating to Oracle](#)
- [Migrating to SQL Server](#)
- [Migrating to an external database](#)
- [Troubleshooting Databases](#)

Step 1. Create a MySQL database

Set up a MySQL database as follows:

- Configure the database to use the InnoDB storage engine
- Create a database on MySQL for FishEye/Crucible to use
- Create a FishEye user on the database
- Configure the database to use utf8 character set encoding
- Configure the database to use utf8_bin collation (to ensure case sensitivity).

Here is an example of how to do that. When FishEye/Crucible and MySQL run on the same machine (accessible through localhost), issue the following commands (replacing fisheyeuser and password with you own values):

```
mysql> SET GLOBAL storage_engine = 'InnoDB';
mysql> CREATE DATABASE fisheye CHARACTER SET utf8 COLLATE utf8_bin;
mysql> GRANT ALL PRIVILEGES ON fisheye.* TO 'fisheyeuser'@'localhost'
IDENTIFIED BY 'password';
mysql> FLUSH PRIVILEGES;
mysql> QUIT
```

For MySQL 5.6 and later, replace the first statement (SET GLOBAL storage_engine = 'InnoDB') with the following:

```
mysql> SET GLOBAL default_storage_engine = 'InnoDB';
```

This creates an empty MySQL database with the name fisheye, and a user that can log in from the host that FishEye is running on who has full access to the newly created database. In particular, the user should be allowed to create and drop tables, indexes and other constraints.

You will also need to set the Server Character set to utf8. This can be done by adding the following in my.ini for Windows or my.cnf for other operating systems (create the file at /etc/my.cnf if it doesn't already exist). It has to be declared in the Server section, which is the section after [mysqld]:

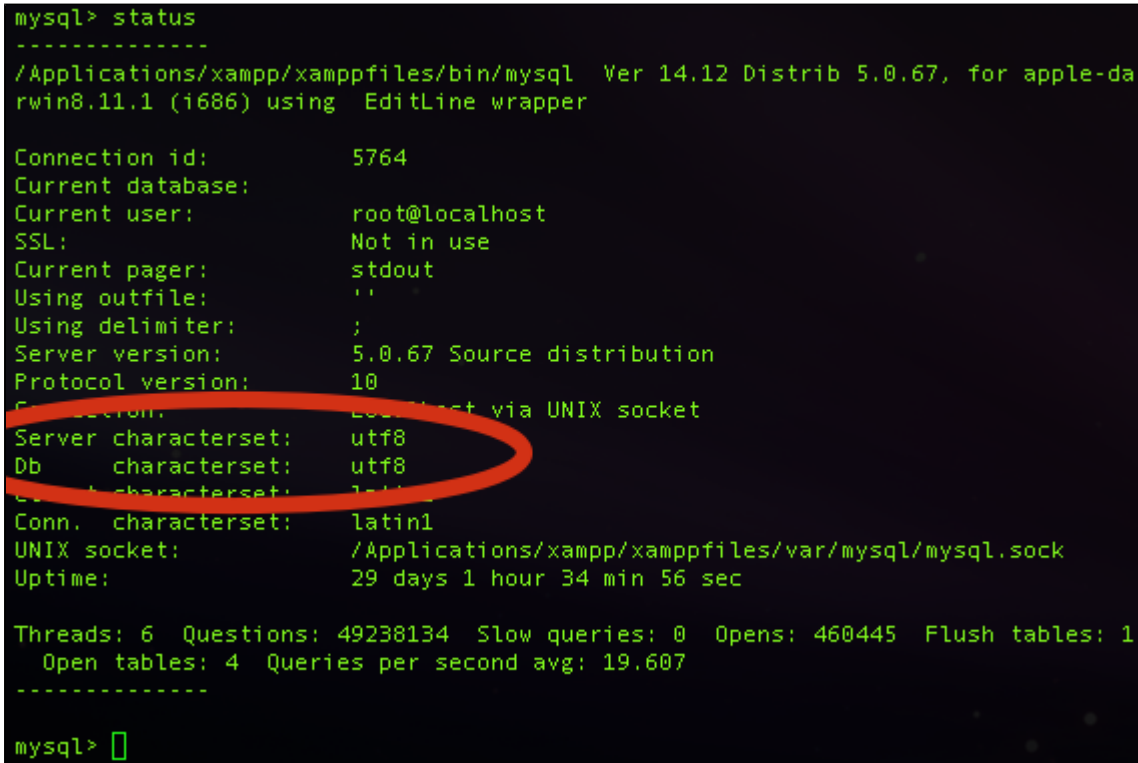
```
[mysqld]
character-set-server=utf8
```

You'll need to restart MySQL for that change to take effect. Now use the status command to verify database

character encoding information:

```
mysql> use fisheye;  
mysql> status;
```

Screenshot: Using the MySQL status command



```
mysql> status  
-----  
/Applications/xampp/xamppfiles/bin/mysql Ver 14.12 Distrib 5.0.67, for apple-da  
rwin8.11.1 (i686) using EditLine wrapper  
  
Connection id:          5764  
Current database:  
Current user:           root@localhost  
SSL:                    Not in use  
Current pager:          stdout  
Using outfile:          ''  
Using delimiter:        ;  
Server version:         5.0.67 Source distribution  
Protocol version:       10  
Connection:             Localhost via UNIX socket  
Server character set:    utf8  
Db character set:        utf8  
Client character set:    latin1  
Conn. character set:     latin1  
UNIX socket:            /Applications/xampp/xamppfiles/var/mysql/mysql.sock  
Uptime:                 29 days 1 hour 34 min 56 sec  
  
Threads: 6 Questions: 49238134 Slow queries: 0 Opens: 460445 Flush tables: 1  
Open tables: 4 Queries per second avg: 19.607  
-----  
mysql>
```

Both the server and database character set must be UTF-8. You may not initially notice a problem because of the [character set overlap](#) but this is indeed a requirement. We've verified that when using a charset such as Swedish 7 bit as the server charset and UTF-8 as the database charset data corruption is easy to reproduce and obvious. This proves that the server character set is as important and that data corruption is possible when there is a charset mismatch. We absolutely cannot suggest a configuration where the server charset and database charset are anything other than UTF-8.

Step 2. Configure FishEye/Crucible to use MySQL, and migrate data

In order to migrate to a different database backend, you must create a backup of sql data, configure the database and finally import the data using a backup restoration process. This can be done from either the FishEye/Crucible administration console, which streamlines the process, or using the command line tool which FishEye/Crucible provides.

Option 1: Migrate using the UI (FishEye/Crucible Administration)

1. Navigate to the Database page in FishEye/Crucible's Administration console.

To log in to the Admin area, you can either:

- click **Administration** at the foot of the page.
- navigate to <http://HOSTNAME:8060/admin/>, where HOSTNAME is the name of the server on which you installed Fisheye.

Once logged in as an [administrator](#) you can also get to the Admin area by clicking the 'cog' menu in the FishEye/Crucible header, and choosing **Administration**.

2. Choose **Edit > Test Connection** to verify that FishEye/Crucible can log in to the database.
3. Select **MySQL** from the database type.
4. Fill in the appropriate fields, replacing the host, port, database name, username and password using the same connection details as used when creating the MySQL database in Step 1 above.
5. Click **Test Connection** to validate the values.

Screenshot: Testing the connection

Database Configuration

To switch to a different database, specify the database's configuration settings in the form below and use the Test Connection button to verify that the database can be used. We are currently connected to the Oracle database at jdbc:oracle:thin:@localhost:1521:xe.

Changes to connection pool settings will not take effect until the application is restarted before the new settings will take effect.

Type: **MySQL** (dropdown menu showing Built-In, PostgreSQL, Oracle, MySQL)

Driver Location: **User Supplied - FISHEYE_INST/lib** (dropdown menu)

URL: **jdbc:mysql://localhost/dbname**

User Name: **username**

Password: *********

Minimum Pool Connections: **5**

Maximum Pool Connections: **20**

Parameters: **useUnicode=true
characterEncoding=UTF8
connectionCollation=UTF8_bin**

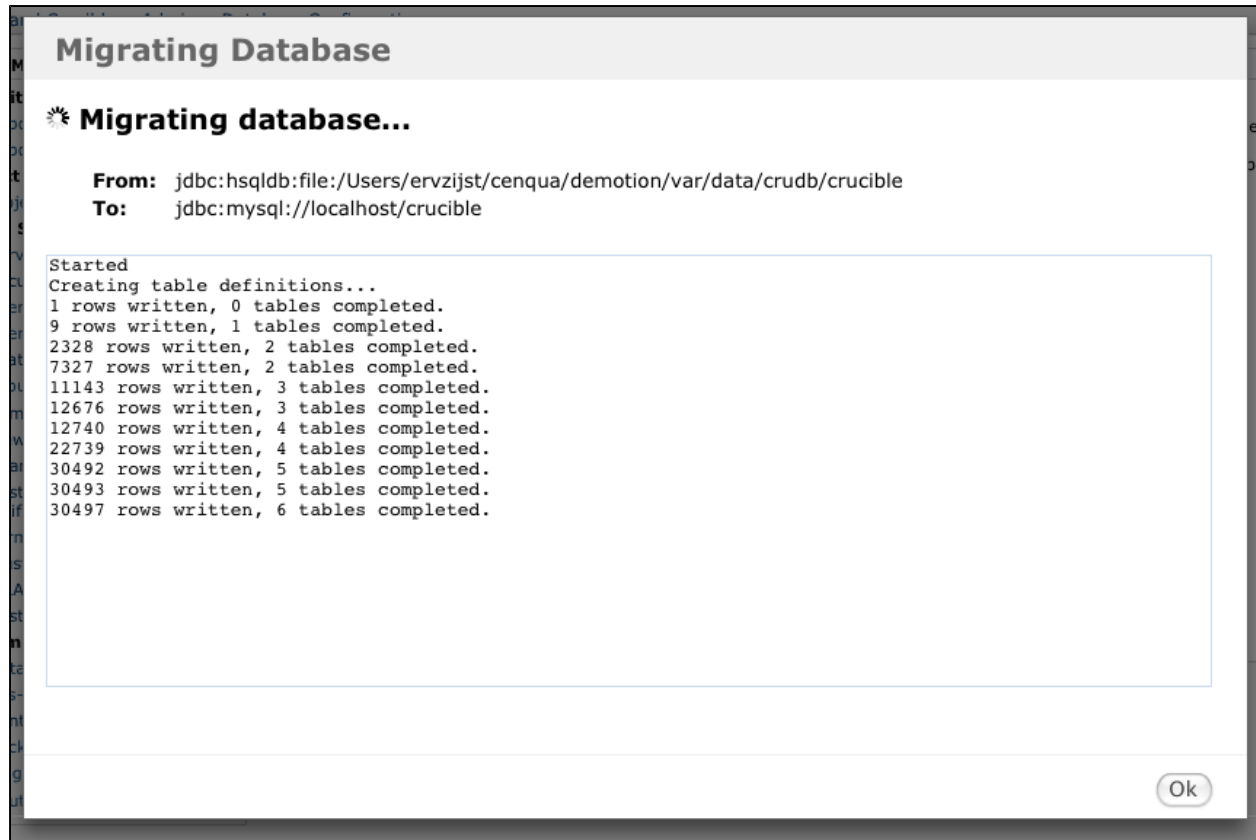
Buttons: **Cancel**, **Save & Migrate**, **Save**, **Test Connection**

If this fails, verify that you have the MySQL JDBC driver `.jar` file in the classpath (see [Prerequisites](#) section above for instructions on how to install the driver). Also, ensure that the database user can log in to the database from the machine that FishEye/Crucible is running on and that all the required privileges are present.

6. Click **Save & Migrate** to start the migration process.

During the migration process (which will take several minutes, depending on the size of your database and network throughput), the product will be inaccessible to users and external API clients. Users will see a maintenance screen that informs them of the process. Should the migration fail for any reason, FishEye/Crucible will not switch to the new database, and will report on the encountered problems. Because the destination database may now contain some, but not yet all data, drop all tables, indexes and constraints before attempting a new migration.

Screenshot: Migrating the database



Option 2: Migrate using the command line

1. Create a backup of the sql data from the FishEye/Crucible instance. Read [Backing up and restoring FishEye data](#) and [Backing up and restoring Crucible data](#) for information on how to create a backup.
2. Run the following command from the <FishEye installation directory>/bin directory:

```
$ ./fisheyectl.sh restore --sql \
    --file /path/to/backup.zip \
    --dbtype mysql \
    --jdbcurl jdbc:mysql://hostname/dbname \
    --username crucible \
    --password password
```

3. When the import is complete, FishEye/Crucible can be started and will use MySQL.

Migrating to Oracle

To switch to an Oracle database, install Oracle and follow the steps below. When they are used together, FishEye and Crucible share the same external database.

Please note that during the migration of database servers, the FishEye/Crucible instance will not be available to users or to external API clients.

Oracle support for FishEye/Crucible and Crucible was introduced in version 2.5.0. In order to migrate to Oracle, your instance must be *currently running* at least version 2.5. If you are running an older version, then you will be required to first upgrade FishEye/Crucible and then migrate.

On this page:

- [Step 1. Install and Create a Oracle Database](#)
- [Step 2. Configure FishEye/Crucible to use Oracle, and Migrate Data](#)

Related pages:

- [Migrating to MySQL](#)
- [Migrating to PostgreSQL](#)
- [Migrating to SQL Server](#)
- [Migrating to an external database](#)
- [Troubleshooting Databases](#)

Step 1. Install and Create a Oracle Database

1. The JDBC drivers for Oracle are bundled with FishEye/Crucible. Skip to step 2 if this meets your needs. If you want to install a specific, different version of the bundled JDBC driver, download the Oracle JDBC driver .jar file from the Oracle website (<http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html>) and copy the .jar file to your FISHEYE_INST/lib directory (create the lib/ directory if it doesn't already exist). Move the existing JDBC .jar file to another location (and back it up). Restart FishEye/Crucible to have it pick up the new driver.
2. Because creating a database with Oracle is a complex process, we recommend speaking to your resident DBA for creation of a new database for usage with FishEye/Crucible. We *highly* recommend installing Oracle with the AL32UTF8 encoding otherwise you may see encoding issues in the product.

Permissions

Ensure the database user has CREATE TABLE, CREATE SEQUENCE and CREATE TRIGGER permissions in addition to the read/write permissions to the database.

Step 2. Configure FishEye/Crucible to use Oracle, and Migrate Data

In order to migrate to a different database backend, you must create a backup of sql data, configure the database and finally import the data via a backup restoration process. This can be done from either the FishEye/Crucible administration console, which streamlines the process, or via the command line tool which FishEye/Crucible provides.

Option 1: Migrate using the UI (FishEye/Crucible Administration)

1. Navigate to the Database page in FishEye/Crucible's Administration console.

To log in to the Admin area, you can either:

- click **Administration** at the foot of the page.
- navigate to <http://HOSTNAME:8060/admin/>, where HOSTNAME is the name of the server on which you installed Fisheye.

Once logged in as an [administrator](#) you can also get to the Admin area by clicking the 'cog' menu in the FishEye/Crucible header, and choosing **Administration**.

2. Then choose **Edit > Test Connection** to verify that FishEye/Crucible can log in to the database.
3. Select Oracle from the database type
4. Fill in the appropriate fields, replacing the host, port, database name, username and password using the same connection details as used when creating the Oracle database in Step 1 above.
5. Click on Test Connection to validate the values

Screenshot: Testing the Connection

Database Configuration

To switch to a different database, specify the database's configuration settings in the form below and use the Test Connection button to verify that the database can be used. We are currently connected to the PostgreSQL database at jdbc:postgresql://localhost:crucible.

Changes to connection pool settings will not take effect until the application is restarted before the new settings will take effect.

Type: Built-In PostgreSQL Oracle MySQL

Driver Location: jdbc:oracle:thin:@localhost:1521:crucible

URL: jdbc:oracle:thin:@localhost:1521:crucible

User Name: crucible

Password: *****

Minimum Pool Connections: 5

Maximum Pool Connections: 20

Parameters:

Cancel Save & Migrate Test Connection Save

If this fails, verify that you have the Oracle JDBC driver .jar file in the classpath (by placing the .jar file in FISHEYE_INST/lib). Also, ensure that the database user can log in to the database from the machine that FishEye/Crucible is running on and that all the required privileges are present.

- Click **Save & Migrate Data** to start the migration process.

During the migration process (which will take several minutes, depending on the size of your database and network throughput), the product will be inaccessible to users and external API clients. Users will see a maintenance screen that informs them of the process. Should the migration fail for any reason, FishEye/Crucible will not switch to the new database and report on the encountered problems. Because the destination database may now contain some, but not yet all data, drop all tables, indexes and constraints before attempting a new migration.

Screenshot: Migrating the Database

Migrating Database

Database migration successful.

From: jdbc:hsqldb:file:/Users/seb/dev/crucible/fe/output/dist_inst/var/data/crddb/crucible
To: jdbc:oracle:thin://localhost:1521/crucible

507 rows written, 31 tables completed.
 511 rows written, 33 tables completed.
 512 rows written, 33 tables completed.
 514 rows written, 34 tables completed.
 529 rows written, 35 tables completed.
 531 rows written, 37 tables completed.
 532 rows written, 37 tables completed.
 532 rows written, 39 tables completed.
 533 rows written, 39 tables completed.
 541 rows written, 40 tables completed.
 543 rows written, 41 tables completed.
 546 rows written, 43 tables completed.
 547 rows written, 43 tables completed.
 558 rows written, 44 tables completed.
 572 rows written, 46 tables completed.
 Adding database constraints...
 Updating database to latest version...
 Switching to new database at: jdbc:oracle:thin://localhost:1521/crucible...
 Complete

Ok

Option 2: Migrate using the command line

1. Create a backup of the **sql data** from the FishEye/Crucible instance. Information on how to create a backup can be found at [Backing up and restoring FishEye data \ Backing up and restoring Crucible data](#)
2. Run the following command from the <FishEye installation directory>/bin directory:

```
$ ./fisheyectl.sh restore --sql \  
    --file /path/to/backup.zip \  
    --dbtype oracle \  
    --jdbcurl jdbc:oracle:thin:@hostname:port:dbname \  
    --username crucible \  
    --password password
```

3. When the import is complete, FishEye/Crucible can be started and will use Oracle.

Migrating to PostgreSQL

To switch to a PostgreSQL database, install PostgreSQL and follow the steps below. When they are used together, FishEye and Crucible share the same external database.

Please note that during the migration of database servers, the FishEye/Crucible instance will not be available to users or to external API clients.

On this page:

- [Step 1. Install and Create a PostgreSQL Database](#)
- [Step 2. Configure FishEye/Crucible to use PostgreSQL, and Migrate Data](#)

Related pages:

- [Migrating to MySQL](#)
- [Migrating to Oracle](#)
- [Migrating to SQL Server](#)
- [Migrating to an external database](#)
- [Troubleshooting Databases](#)

Step 1. Install and Create a PostgreSQL Database

1. The JDBC drivers for PostgreSQL are bundled with FishEye/Crucible. Skip to [Step 2](#) if this meets your needs. If you want to install a specific, different version of the bundled JDBC driver, download the PostgreSQL JDBC driver .jar file from the [PostgreSQL website](#) and copy the .jar file to your FISHEYE_INST/lib directory (create the lib/ directory if it doesn't already exist). Move the existing JDBC .jar file to another location (and back it up). Restart FishEye/Crucible to have it pick up the new driver.
2. Create a new database user (replacing 'username' and 'password' with the appropriate values):

```
$ psql  
> create user username password 'password';
```

3. Create a UTF-8 database and make the newly created user the owner:

```
> create database crucible ENCODING 'UTF-8' OWNER username;
```

4. Make sure the user has full access to the database:

```
> grant all on database crucible to username;
```

Step 2. Configure FishEye/Crucible to use PostgreSQL, and Migrate Data

In order to migrate to a different database backend, you must create a backup of SQL data, configure the database and finally import the data via a backup restoration process. This can be done from either the FishEye/Crucible administration console, which streamlines the process, or via the command line tool which FishEye/Crucible provides.

Option 1: Migrate using the UI (FishEye/Crucible Administration)

1. Navigate to the Database page in FishEye/Crucible's Administration console.

To log in to the Admin area, you can either:

- click **Administration** at the foot of the page.
- navigate to <http://HOSTNAME:8060/admin/>, where HOSTNAME is the name of the server on which you installed FishEye.

Once logged in as an **administrator** you can also get to the Admin area by clicking the 'cog' menu in the FishEye/Crucible header, and choosing **Administration**.

2. Choose **Edit > Test Connection** to verify that FishEye/Crucible can log in to the existing database.
3. Select PostgreSQL from the database **Type**.
4. Fill in the appropriate fields, using the same connection details as used when creating the PostgreSQL database in Step 1 above.
 - a. **Driver Location**: either your own PostgreSQL JDBC or the **Bundled** one that came with FishEye
 - b. **URL**: create this field by replacing the host, port, and database name with your own (i.e. jdbc:postgresql://localhost:5432/<dbname> e.g. jdbc:postgresql://localhost:5432/crucible)
 - c. **Username**: your DB username
 - d. **Password**: your DB password
 - e. **Minimum Pool Connections**: 5 is the default
 - f. **Maximum Pool Connections**: 20 is the default
 - g. **Parameters**: (one per line)
 - i. useUnicode=true
 - ii. characterEncoding=UTF8
5. Click **Test Connection** to validate the values.

Screenshot: Testing the Connection

Database Configuration 🟢

To switch to a different database, specify the database's configuration settings in the form below and use the Test Connection button to verify that the database can be used. We are currently connected to the PostgreSQL database at jdbc:postgresql://localhost:5432/crucible.

Changes to connection pool sizes require a full restart before the new settings will take effect.

Type: **PostgreSQL** (Built-In, Oracle, MySQL)

Driver Location: **Built-In**

URL: jdbc:postgresql://localhost:5432/crucible

User Name: crucible

Password: ••••••••

Minimum Pool Connections: 5

Maximum Pool Connections: 20

Parameters: useUnicode=true
characterEncoding=UTF8

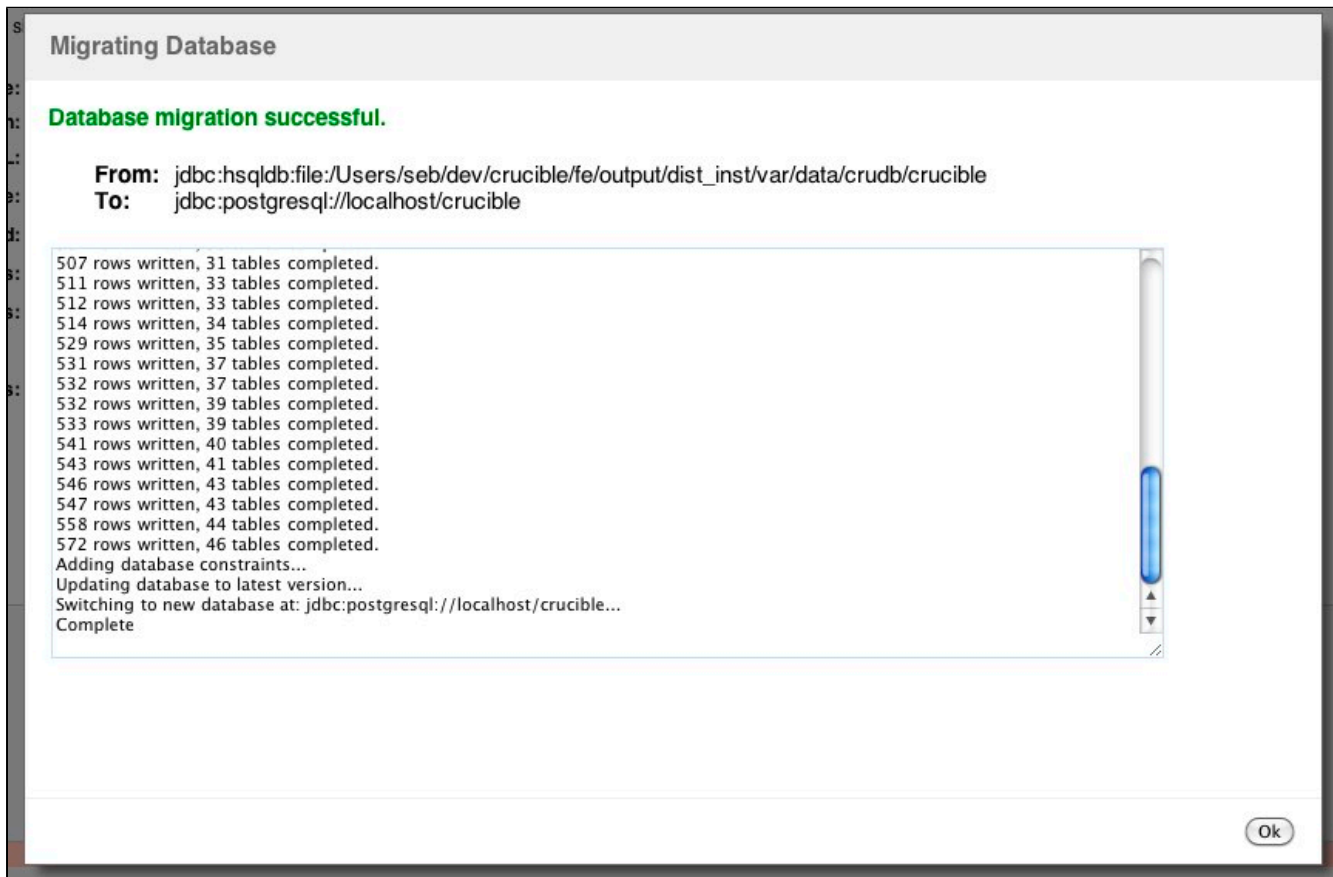
Buttons: Cancel, Save & Migrate, Save, Test Connection

If this fails, verify that you have the PostgreSQL JDBC driver .jar file in the classpath (by placing the .jar file in FISHEYE_INST/lib). Also, ensure that the database user can log in to the database from the machine that FishEye/Crucible is running on and that all the required privileges are present.

6. Click **Save & Migrate** to start the migration process.

During the migration process (which will take several minutes, depending on the size of your database and network throughput), the product will be inaccessible to users and external API clients. Users will see a maintenance screen that informs them of the process. Should the migration fail for any reason, FishEye/Crucible will not switch to the new database and report on the encountered problems. Because the destination database may now contain some, but not yet all data, drop all tables, indexes and constraints before attempting a new migration. If successful, the following message is displayed:

Screenshot: Migrating the Database



Option 2: Migrate using the command line

1. Create a backup of the sql data from the FishEye/Crucible instance. Information on how to create a backup can be found at [Backing up and restoring FishEye data / Backing up and restoring Crucible data](#).
2. Run the following command from the <FishEye installation directory>/bin directory:

```
$ ./fisheyectl.sh restore --sql \
    --file /path/to/backup.zip \
    --dbtype postgresql \
    --jdbcurl jdbc:postgresql://hostname/dbname \
    --username crucible \
    --password password
```

3. When the import is complete, FishEye/Crucible can be started and will use PostgreSQL.

Migrating to SQL Server

To migrate FishEye/Crucible to an SQL Server database, install SQL Server and follow the steps below. When they are used together, FishEye and Crucible share the same external database.

Before you begin

Check that you are using version of SQL Server that is supported for use with FishEye. See [Supported platforms](#).

An existing Java bug prevents connection with Java 1.6.0_29 and above (including Java 1.7.0). Read more about the issue and possible workarounds [here](#).

Step 1. Install and create an SQL Server database

See the [SQL Server Online resources \(MSDN\)](#) for instructions on how to install and create an SQL Server database.

Please note the following FishEye/Crucible-specific information when installing and creating an SQL Server database:

- The JDBC jtds drivers for SQLServer are bundled with FishEye/Crucible. We do not support using the Microsoft distributed jdbc driver.
- The FishEye database user must have permission to connect to the database and to create and populate tables.
- The database user should **not** be the database owner, but **should** be in the db_owner role. (See [SQL Server Startup Errors](#) for details.)
- Your database must be configured to use the Latin1_General_CS_AS collation set.
- Your database should be configured to use snapshot mode for the transaction isolation level. To enable snapshot mode, run:

```
ALTER DATABASE crucible
SET READ_COMMITTED_SNAPSHOT ON;
```

See [this](#) and [this](#) Microsoft MSDN articles for more information.

Note that it is preferable to run the above command after stopping FishEye/Crucible (and with no other applications connected to the SQL Server database), especially if you find that the `alter` statement does not complete quickly.

On this page:

- [Before you begin](#)
- [Step 1. Install and create an SQL Server database](#)
- [Step 2. Configure FishEye/Crucible to use SQL Server and migrate data](#)

Related pages:

- [Migrating to MySQL](#)
- [Migrating to PostgreSQL](#)
- [Migrating to Oracle](#)
- [Migrating to an external database](#)
- [Troubleshooting Databases](#)

Step 2. Configure FishEye/Crucible to use SQL Server and migrate data

In order to migrate to a different database backend, you must create a backup of your SQL data, configure the database and finally import the data via a backup restoration process. This can be done from either the Crucible [administration console](#), which streamlines the process, or via the [command line tool](#) which Crucible provides. These two methods are described below. The following resources may be of interest:

- [Backing up and restoring FishEye data](#)
- [Backing up and restoring Crucible data](#)
- [SQL Server Online resources \(MSDN\)](#)

Option 1: Migrate using the UI (FishEye/Crucible Administration)

Before you begin:

- Note, during the migration process (which will take several minutes, depending on the size of your database and network throughput), **the FishEye/Crucible instance will be inaccessible to users and external API clients**. Users will see a maintenance screen that informs them of the process.
- If you are attempting a migration after a previous migration has failed, you must drop all tables, indexes and constraints before attempting a new migration. This is because the destination database may contain data from the previous migration attempt.
- Verify that you have the jtds JDBC driver .jar file in the classpath (by placing the .jar file in FISHEYE_INST/lib).
- Ensure that the database user can log in to the database from the machine that FishEye/Crucible is running on and that all the required privileges are present.
- If your database is hosted on a SQL Server cluster, you must include the instance name in the JDBC URL

```
jdbc:jtds:sqlserver://<server>[:<port>][/<database>];instance=<instance_name>
```

Please ensure that you use a **SQL Server** user account to log into your database, not a Windows user account.

To configure FishEye/Crucible to use SQL Server and migrate data using the administration console:

1. Navigate to the **Database** page (under 'System Settings') in FishEye/Crucible's Administration console.

To log in to the Admin area, you can either:

- click **Administration** at the foot of the page.
- navigate to `http://HOSTNAME:8060/admin/`, where HOSTNAME is the name of the server on which you installed Fisheye.

Once logged in as an [administrator](#) you can also get to the Admin area by clicking the 'cog' menu in the FishEye/Crucible header, and choosing **Administration**.

2. Configure FishEye/Crucible to use SQL Server, as follows:

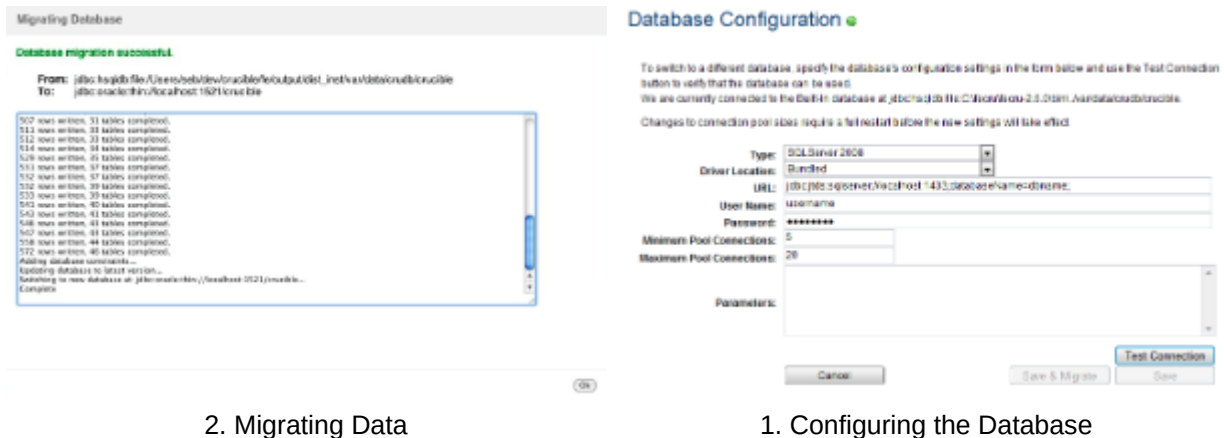
- Select appropriate **SQLServer version** from the **Type** dropdown, matching the version of database you are running.
- Complete the appropriate fields, replacing the **URL** (host, port and database name), **User Name** and **Password** as required, using the same connection details as used when creating the SQL Server database in Step 1 above.

NOTE: The default SQL server instance listens on port 1433. If your instance is not the default, use the port number that is associated with your particular instance.

e.g. **URL**:`jdbc:jtds:sqlserver://localhost:1433;databaseName=your database name here;`

3. Click **Test Connection** to verify that FishEye/Crucible can log in to the database (see 'Testing the Connection' screenshot below).
4. Click **Save & Migrate Data** to start the migration process (see 'Migrating the Database' screenshot below). If the migration fails, FishEye/Crucible will not switch to the new database and will report the problems encountered.

Screenshots: Configuring FishEye/Crucible to use SQL Server and migrating data (click to view full-size images)



2. Migrating Data

1. Configuring the Database

Option 2: Migrate using the command line

To configure FishEye/Crucible to use SQL Server and migrate data using the command line:

1. Create a backup of the sql data from the FishEye/Crucible instance. Information on how to create a backup can be found at [Backing up and restoring FishEye data](#) \ [Backing up and restoring Crucible data](#)
2. Run the following command from the <FishEye installation directory>/bin directory:

```
$ ./fisheyectl.sh restore --sql \
    --file /path/to/backup.zip \
    --dbtype sqlserver2012 \
    --jdbcurl
"jdbc:jtds:sqlserver://hostname:port;databaseName=dbName;" \
    --username crucible \
    --password password
```

3. When the import is complete, FishEye/Crucible can be started and will use SQLServer.

Configuring the database connection pool

This page describes how to change the default database connection pool settings used by FishEye and Crucible.

FishEye and Crucible use the [BoneCP](#) connection pool manager to manage connections to the database. The BoneCP connection pool manager has been preconfigured in FishEye and Crucible to work out-of-the-box for most customers. However, you can override any of the default settings to fine tune performance, if you wish. For example, you may want to try increasing the minimum and/or maximum number of connections, if you are getting significant delays creating connections for clients.

On this page:

- [Setting Minimum and Maximum Database Connections](#)
- [Advanced: Tuning The Connection Pool via Properties](#)

Related pages:

- [Migrating to an external database](#)

Setting Minimum and Maximum Database Connections

To configure the database connection pool:

1. Log in to FishEye or Crucible as an administrator.
2. In the Admin area, click Database (under 'System Settings').
3. Click **Edit**.

4. Enter new values for the **Minimum Pool Connections** and **Maximum Pool Connections** fields.
5. *Advanced users only.* Enter any overrides of BoneCP properties in the **Parameters** field. Enter each parameter-value pair on a new line in the **Parameters** field. See the section below for details of some common properties you may wish to configure.
6. Click **Save**.

Screenshot: Database configuration screen in FishEye / Crucible

Database Configuration

i To switch to a different database, specify the database's configuration settings in the form below and use the *Test Connection* button.

Changes to connection pool sizes require a full restart before the new settings will take effect.

FishEye and Crucible are currently connected to the *SQLServer 2005* database at *jdbc:jtds:sqlserver://fecru-mssql2005-vm.private.atlassian.com:1433;databaseName=crucible;*.

Configuration

Type SQLServer 2005

Driver Location User Supplied - FISHEYE_INST/lib

URL

User Name

Password

Minimum Pool Connections:

Maximum Pool Connections:

Parameters:

Test Connection Save Save & Migrate Cancel

Advanced: Tuning The Connection Pool via Properties

The following table shows some properties you can use to tune the database connection pool used by FishEye and Crucible. You can override any of these parameters by entering parameter-value pairs (on separate lines) into the **Parameters** field of the 'Database Configuration' dialog, as described above.

i Refer to the [BoneCP API javadoc](#) for a full list and description of BoneCP properties.

Parameter	Property and Default Value	Description
Connection timeout	bonecp.idleMaxAgeInMinutes=0	Time to wait before dropping idle connections that idle connections will stay open for
Connection test period	bonecp.idleConnectionTestPeriodInMinutes=60	Database connections are kept alive for this period. This value sets the time

Connection pool partition count	<code>bonecp.partitionCount=3</code>	The database connection pool is split into "partitions" to increase concurrent performance. The number of partitions to use is determined by the <code>bonecp.partitionCount</code> property. Note that if you set this property, but <code>bonecp.minConnectionsPerPartition</code> or <code>bonecp.maxConnectionsPerPartition</code> are set to values below, FishEye will use the "Minimum Pool Connections" and "Maximum Pool Connections" field settings as the minimum and maximum connections calculated to ensure that the field settings are not violated.
Max and min number of connections per partition	<code>bonecp.minConnectionsPerPartition</code> <code>bonecp.maxConnectionsPerPartition</code> Defaults calculated based on value of <code>bonecp.partitionCount</code> and the "Minimum Pool Connections" and "Maximum Pool Connections" field settings	These properties control the minimum and maximum connections <i>in each partition</i>. To calculate the total number of connections in the pool: Total minimum connections = <code>bonecp.minConnectionsPerPartition</code> * <code>bonecp.partitionCount</code> Total maximum connections = <code>bonecp.maxConnectionsPerPartition</code> * <code>bonecp.partitionCount</code> Note that setting these properties overrides the "Minimum Pool Connections" and "Maximum Pool Connections" field settings.
Connection acquire increment	<code>bonecp.acquireIncrement=2</code>	The batch size used when BoneCP acquires connections from the database.
Statement cache size	<code>bonecp.statementsCacheSize=50</code>	The statement cache size defaults to 50 statements per connection. If set to 0, the cache is disabled. For which it defaults to zero cursors .
Connection test query	<code>bonecp.connectionTestStatement</code> By default boneCP will use a metadata request.	The query to send to the DB to maintain connections. This is database specific and should consume the minimal amount of resources. Examples: • MySQL: <code>/* ping */ SELECT 1</code> • PostgreSQL: <code>"SELECT NOW()"</code> If you do not set this, then BoneCP will use <code>SELECT 1</code> instead that should work on all databases. In MySQL, prefixing the statement with <code>/* ping */</code> will use a fast packet instead. See http://blogs.oracle.com/jdev/tips_for_java_developers)

Configuring SMTP

You can configure and customize the FishEye mail server in a number of ways:

- Configure SMTP settings in the FishEye admin area - described [below](#) on this page.
- Customize the templates used for email notifications sent by FishEye - described on [Customizing email notifications](#).
- Tune the mail client in FishEye with system properties - see [JVM system properties](#).

To configure SMTP settings in FishEye:

1. Go to the Admin screen, click **Server** (under 'Global Settings') and look for the 'Mail Server' section.
2. Click **Edit config**.
3. You can edit the following settings:

Send emails from	Server address: Fisheye and Crucible will use the From Address set below. User address: Fisheye and Crucible will use the email address in the user's profile.
From Address	The sender address used when FishEye or Crucible sends an email, e.g. 'fisheye-noreply@example.com'
SMTP Host name	The host name of the SMTP server.
Enable debug	Optional. Turn this on to enable debug logging from the mail server. Useful in tracking down mail server connectivity problems.
SMTP Port	Optional. The port to connect to on the SMTP host. FishEye needs to use port 25 or port 587, because unlike JIRA its initial connection doesn't use SSL. Port 25 will be used if no port is specified.
Use SSL/TLS	Optional, defaults to 'False'. This turns on Secure Sockets Layer/Transport Layer Security security for mail servers that require it, or use it by default.
Username & Password	Optional. Username and password for authenticated SMTP access.
Connection timeout	The time in seconds to wait for a connection to the SMTP server (leave blank for infinite timeout).
Timeout	The time in seconds to wait for a response from the SMTP server (leave blank for infinite timeout).

4. Click **Save**.

Once you have configured SMTP, you can click **Send test email** in the 'Mail Server' section to confirm the SMTP connectivity.

Customizing email notifications

FishEye has a "watch" notification system that sends email notifications to users when commits are detected. Users can opt in for these notifications by "[watching](#)" a particular repository/activity stream.

Email notifications in FishEye can be customized to change their formatting, by editing template files. This page contains instructions for this process.

Editing FishEye email templates

Template files for FishEye are stored in the <FishEye install directory>/template/fisheye and <FishEye install directory>/template/shared folders.

There are sets of templates for both HTML and plain-text emails, as listed in the table below. Note that these templates do not support embedding full diffs into notifications. They are only for changing the appearance and order of certain content inside the messages.

You can edit templates in any text editor.

After an edit, the change to the email template will take place immediately. No restart is required.

If you edit the templates of an operational FishEye instance, you may disrupt notifications that are being sent at that time. To avoid this, shut FishEye down during template editing.

Try and avoid editing the live template file, as FishEye may try to use it while you are editing. This could have unpredictable results. Instead, back up the template file (it's wise to keep original versions of all these files), edit a copy you have made, then overwrite the 'live' template once you have finished.

Advanced editing of FishEye email templates

The email notification templates use the [Freemarker](#) format. Freemarker is a general templating engine enabling automated content.

If you are familiar with Freemarker, more advanced customizations can be made to the email notification templates. However, you make such adjustments at your own risk.

FishEye email template file list

The following template files for FishEye notifications are stored in the <FishEye install directory>/template/fisheye/ folder, or its subfolders and <FishEye install directory>/template/shared/ folder, or its subfolders.

Template filename	HTML	Plain-text
notification-subject.ftl	Shared	
changeset-header.ftl	Shared	
util.ftl	Shared	
changeset-mail.ftl	✓	✓
fisheye-layout.ftl	✓	✓
fisheye-support-request.ftl	✗	✓
password-reset.ftl	✓	✓
plugin-notification.ftl	✓	✓
verify-password-reset.ftl	✓	✓

See also [Customizing Crucible Email Notifications](#).

Freemarker Data Model for Email Templates

Customizing FishEye email templates with Freemarker

This page lists the Freemarker data-model for FishEye email templates. See the [Freemarker documentation](#) for instructions on Freemarker syntax. Use the templates that ship with FishEye as a guide to the properties available on each object.

These templates are used to send both batch (e.g. daily) and immediate emails. The template has access to the **changesets** variable which contains the list of changesets to send.

The default FishEye email templates make use of various data model objects, listed below.

Here is a simple example that prints out each revision in each changeset.

```
[#list changesets as cs]
${cs.id}
Author: ${cs.author}
Comment: ${cs.comment}
Files:
[#list cs.revisionInfos as rev]
  ${rev.path} ${rev.revision}
[/#list]
[/#list]
```

Primary Data Model Objects

Object name	Function	Notes
repname	The name of the repository	
siteurl	Base URL of the FishEye instance	
timezone	The time zone as configured in admin	
watchpath	The path for this watch	
changesets	A list of changesets	

The syntax to use the data model object '**repname**' as an example, is as follows:

```
${repname}
```

Changeset objects

The **changesets** list will contain multiple changesets for batch (e.g. daily) notifications and one element for immediate notifications.

These changeset objects have the following properties:

Object name	Function	Notes
comment	The commit comment	Belongs to a <i>changeset</i>
author	Author of the commit	Belongs to a <i>changeset</i>
dateValue	the date of the commit	Belongs to a <i>changeset</i>
revisionInfos	A list of revisions for the changeset	Belongs to a <i>changeset</i>

For example, to iterate through all the changesets notifications, you would use the following:

```
[#list changesets as cs]
  ${cs.id} ${cs.author}
[/#list]
```

Revision objects

Object name	Function	Notes
path	The path of the file	Belongs to a <i>revisionInfo</i> , under a <i>changeset</i>
revision	The revision number	Belongs to a <i>revisionInfo</i> , under a <i>changeset</i>
binary	Boolean indicating whether file is binary	Belongs to a <i>revisionInfo</i> , under a <i>changeset</i>
dead	Boolean indicating whether file is deleted	Belongs to a <i>revisionInfo</i> , under a <i>changeset</i>
move	Boolean indicating whether file is moved	Belongs to a <i>revisionInfo</i> , under a <i>changeset</i>
copy	Boolean indicating whether file is copied	Belongs to a <i>revisionInfo</i> , under a <i>changeset</i>
added	Boolean indicating whether file is added	Belongs to a <i>revisionInfo</i> , under a <i>changeset</i>

linesAdded	Number of lines added	Belongs to a <i>revisionInfo</i> , under a <i>changeset</i>
linesRemoved	Number of lines removed	Belongs to a <i>revisionInfo</i> , under a <i>changeset</i>

Backing up and restoring FishEye data

FishEye data can be backed up from the admin interface or command line. This page contains the command syntax, options and the required procedure to back up and restore your FishEye instance.

If you are migrating to a different machine, please follow the instructions on our [Migrating FishEye Between Servers](#) page.

Using the admin interface to back up

1. Navigate to the FishEye Admin area (click **Administration** in the footer of any FishEye page).
2. Click **Backup** (under 'System Settings').
3. On the Backup screen, you can edit the name of the backup file that will be generated.
4. Choose the components, described in the table [below](#), that should be included in the backup file. Note that the back up will always include the configuration data (`config.xml`), your license file and the FishEye user data.
 Repository and application caches contain temporary data stored from repository scans and library caches that improve startup time. Both will be recreated automatically by re-scanning the source repositories, so the backup files can be significantly reduced by excluding these (if the cost of re-scanning is acceptable).
5. Click **Create Backup Now**.

On this page:

- [Using the admin interface to back up](#)
- [Using the command line to back up](#)
 - [Advanced backup command line settings](#)
 - [Backup command line examples](#)
 - [Known limitations](#)
- [Scheduling FishEye back ups](#)
- [Restoring FishEye data](#)
 - [Using the command line to restore FishEye data](#)
 - [Restore command line options](#)
- [Migrating backup data](#)

Related pages:

- [Migrating FishEye Between Servers](#)
- [Migrating to an external database](#)
- [Troubleshooting Databases](#)

Component	Purpose
ActiveObjects	Data that is stored by plugins
Repository and application caches	The cache contains data that reflects the state of FishEye's repositories. Without it, FishEye must re-scan its repositories after a backup is restored. The cache also contains OSGI library data that increases startup time. These too can be excluded and will be generated automatically when the application is started.
Plugins	Plugins are 3rd-party extensions that you may have installed, and configuration for all plugins (this includes configuration for Crucible's set of standard plugins).
SQL Database	Refers to the SQL content database (used by both FishEye and Crucible and containing all user profile data, reviews and their comments).

Web templates	In this context, these are custom freemarker templates that you or your users have created. They live in FISHEYE_INST/template.
Uploaded files	In this context, uploads refers to files which are added to Crucible via the web interface (such as patch file reviews). It also includes each repository-backed file that went under review, when Crucible is configured to make a local copy of every reviewed file .

Screenshot: The FishEye Backup screen

Backup ?

 **FishEye and Crucible will be unavailable until the backup completes**

Depending on the number of repositories and reviews, the backup may take a long time to complete and may require large amounts of disk space (especially when you are including the cache). Please make sure you have enough disk space before proceeding.

Create a new backup

Filename

Specify the filename on the server where FishEye and Crucible should export.
This file will be created in /Users/pwatson/fecru-home/backup

Include ☒ ActiveObjects
☐ repository and application caches
☒ plugins and their configuration data
☒ SQL database
☒ web templates
☒ uploaded files and local copies of files under review

Choose the components that FishEye and Crucible should include in the backup.

or [Manage Scheduled Backups](#)

Using the command line to back up

i Your FishEye instance must be running during the back up.

1. Open a terminal or command prompt on the FishEye server computer.
2. Navigate to <FishEye install directory>/bin/.
3. Run the backup command on the command line using any desired options (described in the table [below](#)).
By default, all backup components, except repository cache data, is included in the backup.

The backup is created as a new Zip archive file and placed in the FISHEYE_INST/backup/ directory.

i Note that if your FishEye instance uses a custom FISHEYE_INST directory, make sure the environment variable is properly set when running the backup command.

Option	Switch	Default setting
Quiet mode	-q, --quiet	No
Output filename	-f, --file	FISHEYE_INST/backup/ is the default directory.
Compression level	-c, --compression	Yes (6)
Anonymize	-a, --anonymize	No
Cache Backup	--cache	No. By default, the cache data is excluded from backups.

Advanced backup command line settings

In some cases it might be preferable to only back up a limited set of items. This could be useful when your instance uses an external database such as MySQL or PostgreSQL and your DBA has already configured automatic back ups in the database. The commands below allow this.

Option	Switch	Description	Default
Exclude Plugins	--no-plugins	Excludes plugins from the backup.	No. By default, plugins are included in every backup.
Exclude Templates	--no-templates	Excludes templates from the backup.	No. By default, templates are included in every backup.
Exclude Uploads	--no-uploads	Excludes uploaded files (such as patch reviews, stored in Crucible's internal database) from the backup.  This item only applies when using Crucible with FishEye.	No. By default, uploads are included in every backup.
Exclude SQL Database	--no-sql	Excludes the SQL content database used by both FishEye and Crucible.	No. By default, this data is included in every backup.

Backup command line examples

These examples are for use in a Linux-like operating system. When using these commands on Windows, use the filename `fisheyectl.bat` and use the correct slashes. Run the command from the `<FishEye install directory>/bin/` directory.

The basic syntax of the backup command is as follows:

```
$ ./fisheyectl.sh backup [OPTIONS]
```

To see inline help for all backup options, run the following command in the `<FishEye install directory>/bin/` directory:

```
$ ./fisheyectl.sh backup --help
```

Backing up with compression of 9, quiet mode and setting an output location:

```
$ ./fisheyectl.sh backup --compression 9 -q -f
/application_backups/fisheye/20090215.zip
```

Back up including cache data (also includes all default components):

```
$ ./fisheyectl.sh backup --cache
```

Known limitations

Please note that the limitations below are common to any Java-based backup tool.

Archives Containing Over 65535 Files

Versions of Java earlier than v1.6 (b25) are incapable of handling zip files that contain more than 65,535 files. The solution for this problem is to either upgrade to a version of Java later than v1.6 (b25), or ensure that the archive does not exceed the threshold (contains less than 65,535 files). The FishEye cache (not included in backups by default) can be a contributor of many small files. Hence, exclude the cache from backups if this is likely to be a concern.

Archives Larger Than 4GB

Java has trouble reading and writing zip files that are larger than 4GB. As of release 1.5 Java appears capable of reliably creating archives that are over 4GB, but remains unable to extract them. For details see [Sun's bug report](#). Also be aware of the fact that some file systems (including [FAT32](#)) have trouble with files larger than 4GB.

As a workaround, make sure you do not create archives that are larger than 4GB. The FishEye cache (not included in backups by default) can be a contributor of a lot of small files (although these tend to compress very well). If you still want to archive everything and end up with an archive that is too large, consider creating separate backups for the FishEye cache and uploaded files respectively.

Scheduling FishEye back ups

To set a schedule for automatic back ups:

1. Go to the Admin area and click **Backup** (under 'System Settings').
2. Click **Manage Scheduled Backups** at the bottom of the page.
3. Click **Edit**, set the desired options, then click **Save**.

The options for scheduled back ups are detailed in the table below:

Option name	Description	Allowed Values
Disable Scheduled Backups	Stops regular back ups from taking place.	On (disabled) or Off (enabled)

Backup path	The path where the backup .zip file will be stored.	Any system or network path that FishEye or Crucible can access. This cannot be changed using the FishEye interface. Please edit your <code>config.xml</code>, and add the <code><path></code> attribute according to this example: <pre> <backup> <schedule enabled="true"> <path>path_to_backup_location</path> <prefix>fisheye_backup_</prefix> <date-pattern>yyyy_MM_dd</date-pattern> <frequency>WEEKLY</frequency> <time>07:00</time> <items> <item name="sql"/> <item name="templates"/> <item name="cache"/> <item name="ao"/> <item name="plugins"/> <item name="uploads"/> </items> </schedule> </backup> </pre> Save changes, and restart your FishEye instance.
Backup file prefix	Characters that will be added to the beginning of the backup file name.	Any string of characters that can be used as part of a filename on the local operating system.
Backup file date pattern	Sets a date for the next (or initial) back up to take place.	Any valid date in the format <code>yyyy_MM_dd</code> (year, month, day of the month).
Backup frequency	Sets how often the back up will take place.	Can be set to every day , every Sunday , Monday to Friday and first day of the month .
Backup time (HH:mm)	The time when the back up will take place.	Any valid 24-hour time in the format <code>HH:mm</code> (hours, minutes).
Include	Specifies which items must be included in the backups (these components are explained at the top of this page).	As per the options for regular on-demand back up (These components are explained at the top of this page).

Screenshot: Scheduling back ups in FishEye

Scheduled Backups

 Please choose the schedule for your backups carefully. FishEye and Crucible will be unavailable for the duration of the backup process.

 Your managed repositories will not be backed up!
FishEye does not automatically backup content stored in your managed repositories. Please ensure that a backup process which is satisfactory to your requirements is in place. [Read more](#)

Schedule backups

Disable scheduled backups ☐ Switch scheduled backups on and off.

Backup directory The location to store the backups. Edit config.xml to change this setting.

Backup file prefix The prefix for the generated backup file.

Backup file date pattern The date/time format you specify will be added to the file prefix to differentiate between backup files.

Backup frequency How often do you want FishEye and Crucible to run a backup.

Backup Time (HH:mm) What time would you like FishEye and Crucible to run the backup, e.g. 17:30.
Note that this is 24 hour notation in local server time. Current local server time is: 02:04.

Include ☐ ActiveObjects
☒ repository and application caches
☒ plugins and their configuration data
☒ SQL database
☒ web templates
☒ uploaded files and local copies of files under review
Choose the components that FishEye and Crucible should include in the backups.

 Be aware that scheduled back ups can fill up disks unless you regularly move or delete old archives.

Restoring FishEye data

Using the command line to restore FishEye data

There is currently no way to restore a backup from the admin interface because FishEye must be shut down during a data restore.

 Restoring a backup will irreversibly overwrite the data of your installation with the data from the backup archive. Note that you cannot restore data into versions of FishEye which are older than the version that created the backup.

1. [Install FishEye](#) into a new, empty directory (this must be the same version that the backup was created from, or later).
2. Make sure the FishEye instance is not running.
3. Open a command line interface on the FishEye server computer.
4. Run the restore command on the command line with any desired options.
5. The specified elements will be restored.
6. Start the FishEye instance.
7. When using FishEye integrated with Crucible, you will need to re-index your repositories after restoring data, unless the backup archive was created with the `--cache` option.

Restore command line options

By default, the restore process will restore all items found in the backup archive (so if you included the caches using the `--cache` option, these will automatically be restored). However, you can specify a partial restore, by explicitly specifying the item names on the command line.

If you are using an external database (as opposed to the default HSQL database), make sure the JDBC driver file is present in the `FISHEYE_INST/lib` directory when running restore.

Furthermore, if you are restoring to a new FishEye instance and install directory, and select `--dbtype` of `mysql`, you must download the JDBC driver and accept the license agreement before proceeding with the restore.

The options available for use with the restore command are listed in the following table:

Option	Switch	Description
Choose file to restore from	-f PATH/FILENAME, --file PATH/FILENAME	(Required) Restore the backup from PATH/FILENAME.
Repository and application caches	--cache	Restore the repository cache backup.
Plugins	--plugins	Restore 3rd-party plugins and their configuration data.
Web templates	--templates	Restore freemarker templates from the backup (the restored instance will use the built-in templates).
Uploaded files	--uploads	Restore uploads (e.g. patch files uploaded into Crucible and contents of files under review).  This item only applies when using Crucible with FishEye.
SQL Database	--sql	Restore the SQL database containing user profiles, reviews and review comments.
ActiveObjects	--ao	Restore ActiveObjects data stored by plugins.
List backup contents	-l, --list	List the contents of the backup file, and exit.
Set database type	-t, --dbtype	SQL database type (built-in, mysql, oracle, postgresql, sqlserver2005, sqlserver2008, sqlserver2012). Only required when restoring to a database location different to that used at back up time.
Set JDBC URL	-j, --jdbcurl	JDBC URL of the SQL database. Only required when restoring to a database location different to that used at used at back up time (not applicable for built-in).
Set JDBC username	-u, --username	JDBC username of the SQL database. Only required when restoring to a database location different to that used at used at back up time (not applicable for built-in).
JDBC password	-p, --password	JDBC password of the SQL database. Only required when restoring to a database location different to that used at used at back up time (not applicable for built-in).
JDBC class	-d, --driver	Optionally, specify the JDBC driver class name needed to access the SQL database. Only required when restoring to a database location different to that used at used at back up time and when using a different JDBC driver than the standard driver associated with the database specified though - -dbtype. (Not applicable for built-in.)
JDBC driver source	-s, --driver-source	Optionally, specify the JDBC driver file. The default ('bundled') is to use the bundled JDBC driver file; 'user' = use the driver from the FISHEYE_INST/lib directory.
Suppress output	-q, --quiet	Suppress the output messages from the restore program on the command line.
Overwrite the existing DB	--force	Overwrite the existing database, without warning.

Don't batch SQL inserts	<code>--no-batch-sql</code>	Do not batch SQL inserts. This can be useful when diagnosing errors.
Display the help	<code>-h, --help</code>	Display the help, and exit.

i These examples are for use in a Linux-like operating system. When using these commands on Windows, use the filename `fisheyectl.bat` and use the correct slashes. Run the command from the `<FishEye install directory>/bin/` directory.

The basic syntax of the restore command is as follows:

```
$ ./fisheyectl.sh restore -f /path/to/backup_2009-10-02_1138.zip
[OPTIONS]
```

To see inline help for all backup options, run the following command in the `<FishEye install directory>/bin/` directory:

```
$ ./fisheyectl.sh restore --help
```

Restoring a backup with cache data (also restores all default components):

```
$ ./fisheyectl.sh restore --cache
```

Migrating backup data

When the process restores a SQL database, it looks at the configuration data (`config.xml`) included in the backup archive to learn which database product was used and how to connect to it. When FishEye uses the built-in HSQLDB database (which is the default), the restored instance will also use that.

However, when the restored instance will use a different database than the backed up instance (for instance, HSQLDB was used at the time the backup was created, but it needs to be restored on MySQL), you should use the command line options to point the process to the new database which must exist before running the restore.

The restore will only populate the database, it will not create the database.

Command line example: migrating backup data to MySQL

i These examples are for use in a Linux-like operating system. When using these commands on Windows, use the filename `fisheyectl.bat` and use the correct slashes. Run the command from the `<FishEye install directory>/bin/` directory.

Restoring to a FishEye instance that uses a different database (ensure the MySQL driver jar file is present in the `FISHEYE_INST/lib` directory)

```
$ ./fisheyectl.sh restore \
--username john \
--password smith \
--jdbcurl "jdbc:mysql://localhost:3306/crucible;" \
--dbtype mysql \
--file /path/to/backup_2009-10-02_1138.zip
```

Managing add-ons

An add-on is an installable component that supplements or enhances the functionality of FishEye in some way. For example, the [FishEye Release Report](#) is a free add-on that presents a release report of the new commits in a branch for a given release (based on the Commit Graph), including all commits in feature branches that were merged to the release branch.

FishEye comes with many pre-installed add-ons (called system add-ons). You can install more add-ons, either by acquiring the add-on from the [Atlassian Marketplace](#) or by uploading it from your file system. This means that you can install add-ons that you have developed yourself. For information about developing your own add-ons for FishEye, see the [FishEye developer's guide](#).

On this page:

- [About the Universal Plugin Manager \(UPM\)](#)
- [Administering Add-ons in FishEye](#)

You may notice that the terms 'add-on' and 'plugin' both appear in the Atlassian documentation and tools. While the terms are often used interchangeably, there is a difference. A plugin is a type of add-on that can be installed into an Atlassian host application. Plugins are what developers create with the Atlassian SDK. But there are other types of add-ons as well. For example, the JIRA client is an add-on that runs as a separate program rather than as a plugin to JIRA. This documentation uses the term 'add-on' most often.

About the Universal Plugin Manager (UPM)

You administer add-ons for FishEye using the Universal Plugin Manager (UPM). The UPM is itself an add-on that exposes add-on administration pages in the FishEye Administration Console. UPM works across Atlassian applications, providing a consistent interface for administering add-ons in FishEye, Crucible, Confluence, JIRA, Bitbucket Server and Bamboo.

UPM comes pre-installed in recent versions of all Atlassian applications, so you do not normally need to install it yourself. However, like other add-ons, the UPM software is subject to regular software updates. Before administering add-ons in FishEye, therefore, you should [verify your version](#) of the UPM and update it if needed.

Administering Add-ons in FishEye

You can update UPM, or any add-on, from the UPM's own add-on administration pages. Additionally, you can perform these tasks from the UPM administration pages:

- Install or remove add-ons
- Configure add-on settings
- Discover and install new add-ons from the [Atlassian Marketplace](#)
- Enable or disable add-ons and their component modules

For information on performing these add-on administration tasks, see the [Universal Plugin Manager documentation](#).

Linking FishEye to JIRA

JIRA Software is Atlassian's issue-tracking and project-management application. You can connect FishEye to JIRA Software for one or both of the following:

- Integrations that [share data and and functionality](#).
- Delegated FishEye [user and group management](#) in JIRA.

Configuring those two aspects are separate processes, and are described on this page. Note that you can link FishEye with one or more instances of JIRA Software. If you're linking FishEye to an Atlassian Cloud JIRA instance, please see [Link to server applications from Cloud](#).

On this page:

- [Initial configuration](#)
- [Quick setup with the FishEye Setup Wizard](#)
- [Integrate with JIRA Software](#)
- [Use JIRA Software to manage FishEye users](#)
- [Project links](#)
- [Notes](#)

Your user tiers don't need to match between JIRA Software and FishEye/Crucible in order to integrate them. JIRA Software users that are not FishEye users will see the same view as FishEye users within JIRA Software, but will not be able to log in to FishEye to view the source/reviews.

Initial configuration

You should check that the instances of JIRA Software that you wish to link to FishEye are configured to accept remote API calls; in each JIRA Software instance, go to **Administration > System > General Configuration** and turn on **Accept remote API calls** (under 'Options'). A restart is not required. See [options](#) for more details.

Quick setup with the FishEye Setup Wizard

When you are installing FishEye, the setup wizard allows you to configure the JIRA connection automatically. This is a quick way of setting up your integration with JIRA for user management. See [Configuring JIRA integration in the Setup Wizard](#).

If you did not set up JIRA integration during FishEye setup at install time, you can still configure that from the FishEye administration area as described below.

Integrate with JIRA Software

See [JIRA Integration in FishEye](#) for an overview of the benefits you get when FishEye and JIRA are integrated.

See [Linking to another application](#) for details about linking FishEye and JIRA servers to get those integration benefits.

See also [Enabling Smart Commits](#).

Use JIRA Software to manage FishEye users

See [Connecting to JIRA for user management](#).

Project links

We do not recommend the use of project links with FishEye 2.9 and later, if you have [JIRA 5.0 or later](#) as well as the latest version of the [JIRA FishEye Plugin](#). This is because application links now provide all of the functionality previously available with project links. However, project links are retained for the following reasons:

- Setting up project links provides a way to restrict the scope of JIRA searches to a specific FishEye repository.
- Legacy configurations can continue to use project links without any need for changes.
- Third-party plugins may continue to rely on project links for their functionality.

Notes

- **JIRA requires manual refresh of the FishEye repository cache when repository changes are made** — When a repository is removed, or when there has been any change in FishEye repositories, JIRA does not update the FishEye repository list cache automatically. You must manually refresh the repository list

cache. This is done in JIRA: go to **Administration > FishEye Configuration** and click **Refresh Cache** (next to 'Repository List Cache').

- **FishEye doesn't check for invalid JIRA issue keys** – FishEye doesn't check for invalid issue keys, such as 'UTF-8'. An error will result if FishEye tries to connect to an issue that doesn't exist.
- **FishEye doesn't recognize custom JIRA issue key formats** – FishEye assumes that JIRA issue keys are of the default format for JIRA 5.0 and later: that is, up to 10 letters, followed by a hyphen and the issue number (i.e. [a-zA-Z]{10}-[0-9]+), for example FishEye-123.
- **If FishEye/Crucible will be running on the same machine as JIRA** (already installed), you need to ensure that the URL paths are different for FishEye/Crucible and JIRA Software. Change the default FishEye/Crucible path as follows:
 - Edit the `config.xml` in your FishEye/Crucible data directory.
 - Add the context attribute to the `web-server` element:

```
<web-server context="/fisheye">
```

JIRA FishEye Plugin version compatibility

Compatibility matrix for JIRA and the FishEye Plugin... WIP

Which integration features depend on the FEP?

JIRA version	Plugin version	FishEye/Crucible version
5.2 – 5.2.11	5.2.8	
5.0 – 5.1.8	5.0.10	2.4.1 – 2.8.2
5.0 – 5.1.12	5.0.4.1	2.4.1 – 2.7.15
5.0 – 5.0.7	5.0.0+	2.4.1 – 2.7.15

All versions of the plugin can be downloaded from <https://maven.atlassian.com/content/repositories/atlassian-contrib/com/atlassian/jira/plugins/jira-fisheye-plugin/>

Linking to another application

Linking two Atlassian applications, for example FishEye and JIRA Software, allows you to access one application's functions and resources from within the other. You link two applications by creating an application link between them.

See [Integrating FishEye with Atlassian applications](#) for an overview of the benefits of integrating your Atlassian applications with FishEye.

When linking an [Atlassian Cloud](#) instance to FishEye, your FishEye application must be accessible through port 80 (or port 443 with a valid (CA) SSL certificate). See these pages for more information:

- [Restricted functions in JIRA Cloud applications](#)
- [Link to server apps from Cloud](#)

To link two Atlassian applications:

1. Click the 'cog' menu in the FishEye header, and choose **Administration** (you'll need to be logged in as an [administrator](#) to see this link).
2. Choose **Application Links** in the left-hand panel. The Application Links configuration page displays links that have already been set up.
3. Enter the URL of the application you want to link to, then click **Create new link**.
 - If you check **The servers have the same set of users...** then this link will be configured using OAuth (with impersonation) authentication.
 - If you are *not* an admin on both servers you won't be able to set up a 2-way (reciprocal) application link. If you want to go ahead and create a 1-way link anyway, clear the **I am an administrator on both instances** checkbox.
4. Use the wizard to finish configuring the link.

Note that:

- For FishEye 3.2 and later, creating a new application link now uses OAuth by default and enables both 3-legged OAuth (3LO) and 2-legged OAuth (2LO).
- When updating older application links (that perhaps used Trusted Apps authentication) to use OAuth, 3LO is used by default, but you need to explicitly enable 2LO using the check box in the application link configuration settings.
- Only use the 2LO with impersonation option in the application link configuration settings if your servers both have the same set of users and they fully trust each other.

When you complete the wizard, the Application Links plugin will create the link between your applications using the most secure authentication method that is supported between the two application types. After the link has been set up, it will appear on the "Configure Application Links" page. You can use this page to edit the configuration of application links to make them more secure or to change the link settings:

- To edit the settings of the application link (for example, to [change the authentication type](#) of the link), select **Edit**.
- If you've set up multiple links to the same type of application (for example, multiple JIRA servers), you can use the **Make Primary** link to specify which application is the default instance. See [Making a Primary Link for Links to the Same Application Type](#) for more information.

Having trouble integrating your Atlassian products with application links?

We've developed a [guide to troubleshooting application links](#), to help you out. Take a look at it if you need a hand getting around any errors or roadblocks with setting up application links.

System settings

- [FishEye folder layout](#)
- [Blame calculations](#)
- [Command-line options](#)
- [Configuring web hooks](#)
- [Enabling Smart Commits](#)
- [Configuring commit hooks](#)
- [Configuring ViewVC compatibility](#)
- [Customizing the front page](#)
- [Configuring raw file download settings](#)
- [Software update notifications](#)
- [Enabling Access Logging in FishEye](#)
- [FishEye and Crucible Cookies](#)
- [Configuring the FishEye web server](#)
- [Running FishEye as a Windows service](#)
- [Configuring indexing](#)
- [Optimizing the Search Indexes on Restart](#)
- [Customising FishEye's look and feel](#)
- [Environment variables](#)
- [Configuring outgoing web proxy support for FishEye or Crucible](#)
- [Using AJP with FishEye](#)

- [Custom authentication](#)
- [Configuring SSL cipher suites for Jetty](#)
- [JVM system properties](#)
- [Configuring user managed mappings](#)

FishEye folder layout

By default, FishEye will create a self-contained instance directory within the `<FishEye install directory>` – the directory where you unzip the package. The FishEye instance directory is where your FishEye data is stored.

For a production environment, you *should not* locate your FishEye instance directory inside the `<FishEye install directory>` — they should be entirely separate locations. If you do put the instance directory in the `<FishEye install directory>` it will be overwritten, and lost, when FishEye gets upgraded. And by the way, you'll need separate FishEye instance directories if you want to run multiple copies of FishEye.

You should create your FishEye instance directory, and then tell FishEye where you created it by setting a `FISH EYE_INST` environment variable. See [Installing FishEye on Windows](#) and [Installing FishEye on Linux and Mac](#) for information about setting the `FISHEYE_INST` environment variable.

Furthermore, we recommend that the instance directory be [secured against unauthorized access](#).

On this page:

- [Default FishEye folder layout](#)
- [Recommended FishEye folder layout](#)

Default FishEye folder layout

The default FishEye folder layout includes the FishEye instance data inside the `<FishEye install directory>`. This layout is not recommended for production environments.

<code><FishEye install directory>/conf/onfig.xml</code>	Configuration file.
<code><FishEye install directory>/var/</code>	Directory under which FishEye stores its data.
<code><FishEye install directory>/var/data/</code>	Persistent data.
<code><FishEye install directory>/var/cache/</code>	Caches and indexes.
<code><FishEye install directory>/var/log/</code>	Log files.
<code><FishEye install directory>/var/tmp/</code>	Temporary files.
<code><FishEye install directory>/cache/</code>	Caches and indexes. (and also in <code><FishEye install directory>/var/cache</code>)
<code><FishEye install directory>/bin/</code>	Scripts for controlling FishEye.
<code><FishEye install directory>/lib/</code>	FishEye's dependent libraries.
<code><FishEye install directory>/syntax/</code>	Syntax highlighting definitions.

<code><FishEye install directory>/ ...</code>	Remainder omitted for brevity.

Recommended FishEye folder layout

The FishEye folder layout that we recommend for production environments has the FishEye instance directory, which contains your instance data and is defined by `FISHEYE_INST`, in a completely separate location from the `<FishEye install directory>`.

<code>\$FISHEYE_INST/config.xml</code>	Configuration file.
<code>\$FISHEYE_INST/var/</code>	All persistent and most temporary data.
<code>\$FISHEYE_INST/cache/</code>	Caches and indexes (and also in <code>\$FISHEYE_INST/var/cache</code>)
<code>\$FISHEYE_INST/lib/</code>	Site-specific Java libraries (.jars) that FishEye should load on startup. (Do not copy the dependent <code><FishEye install directory>/lib/</code> files into here.)
<code>\$FISHEYE_INST/syntax/</code>	Site-specific syntax highlighting definitions.
<code>\$FISHEYE_INST/system.properties</code>	Used for setting system properties within the FishEye JVM, which may sometimes be useful for support purposes. (Note: the other way to set properties is using <code>FISHEYE_OPTS</code> , e.g.: <code>export FISHEYE_OPTS=-Dpropname=value</code>)
<code><FishEye install directory>/lib/</code>	FishEye's dependent libraries.
<code><FishEye install directory>/syntax/</code>	FishEye bundled highlighting definitions.
<code><FishEye install directory>/bin/</code>	
<code><FishEye install directory>/ ...</code>	Remaining files are found under <code><FishEye install directory>/</code> .

Blame calculations

Blames are the annotations available when viewing source files or diffs (in both FishEye and Crucible) that allow you to determine who last modified a line of code and in which revision:

Blame calculation

FishEye and Crucible calculate blames from the internal repository cache during indexing time. On rare occasions that may not be possible. In such cases FishEye and Crucible have a mechanism to retrieve blame data directly from the repository, either during indexing time, or when requested by browsing source files and diffs. However, as contacting the repository can be an expensive operation and may noticeably affect indexing time and/or responsiveness, this mechanism is off by default for both cases since 3.10.

Here are the system properties controlling blame calculations; the default values should be appropriate for most cases:

System property	Description
<code>fisheye.blame.calc.enable</code>	Controls whether blames are calculated during indexing. This property is enabled by default. Disabling it means blames are not calculated during indexing and will be missing (or could optionally be retrieved at request time). Note that blame calculation during indexing is disabled if the Store Diff Info setting is disabled. See Store diff info.
<code>fisheye.blame.scm.fallback.enable</code>	Controls whether blame data is retrieved from the SCM during indexing (when blame couldn't be based on ancestors). This property is disabled by default to avoid performance overhead. This means some indexes may lack blames. In such cases, the blames can either be fetched from SCM at request time (if <code>fisheye.enable.request.blame.calculation</code> = true) or they will be missing from the source file and diff views. Enabling this property means SCM will be contacted (possibly multiple times) during indexing. This increases the indexing time, but ensures all blames are readily available at request time.
<code>fisheye.enable.request.blame.calculation</code>	Controls whether users can request blame data from the SCM when viewing the file or diff (when blame couldn't be calculated or retrieved during indexing). This property is disabled by default. Enabling it means for all missing blames FishEye and Crucible will retrieve blames from the repository at request time.

Missing blame

When blame data is not available, the Author and Revision columns display "n/a" like this:

developerX	1452972	36	<document xmlns="http://maven.apache.org
		37	xmlns:xsi="http://www.w3.org/2001/X
		38	xsi:schemaLocation="http://maven.ap
n/a	n/a	39	
		40	<properties>
		41	<title>Release Notes</title>
		42	<author email="martinc@apache.org">Ma
		43	</properties>
		44	
		45	<body>
developerX	1454695	46	<release version="1.3" description="r

Blames may be not available:

- When indexing of a particular revision has not yet completed. This is a temporary state and blames should appear when indexing has finished.
- When the blame calculation failed during indexing and the `fisheye.enable.request.blame.calculation` is set to false (the default value).
- If the repository configuration property **Store Diff Info** setting is disabled. See [Store diff info](#).

Command-line options

A FishEye instance can be managed using the `fisheyectl` script. Before running this script you need to ensure that you have set the `JAVA_HOME` environment variable, or that `java` is on the path.

Unix usage:

```
<FishEye install directory>/bin/fisheyectl.sh c
ommand [options]
```

Windows usage:

```
<FishEye install directory>\bin\fisheyectl.bat
command [options]
```

The `command` parameter can be one of `run`, `start` or `stop` (see below). You can also find convenience scripts for running each of these commands, such as `run.sh` or `run.bat`.

On this page:

- [run](#)
- [start](#)
- [stop](#)
- [fullscan](#)
- [rescan](#)
- [reindex](#)
- [scannow](#)
- [backup / restore](#)
- [svnrules](#)
- [Read more](#)

run

The `run` command starts FishEye. This command runs FishEye in the foreground. It does not fork a background process.

start

The `start` command has the same options as `run`, but starts FishEye in the background.

Windows: FishEye will be run in a separate `cmd.exe` window.

Unix: FishEye will be run with `nohup` and the console output will be redirected to `$FISHEYE_INST/var/log/fisheye.out`.

Options (for both `run` and `start`):

<code>--config path</code>	Load configuration from the file at path. Default is <code>\$FISHEYE_INST/config.xml</code> .
<code>--quiet</code>	Do not print anything to the console.
<code>--debug</code>	Print extra information to the debug log.
<code>--debug-perf</code>	This option is redundant and is now a synonym for <code>--debug</code> .

The following `--X` prefixed options are currently available, but are typically used only for specific workarounds and edge cases in support scenarios. They may be removed or changed in any future release:

<code>--Xtab-width nchars</code>	Specifies the number of spaces to use to represent a tab character. The default is 8.

<code>--Xdisable-dirtree-empty-checks</code>	When rendering the directory tree on some pages, FishEye calculates if each directory subtree is empty. For massive repositories, this calculation can cause the page to take a long time to render. This option disables the calculation that determines emptiness. If you start FishEye with this flag, then empty directories will no longer show up as grey (i.e. deleted) and the option to hide them will also disappear.
<code>--Xdisable-content-indexing</code>	Disables the generation of a full-text index for file content. This prevents further indexing, but does not delete any existing full-text indexes. FishEye will not warn you if you specify this option but still try to do a content search. This option is useful if you do not use content search and you are finding FishEye is taking a long time to index your content.
<code>--Xno-slurp</code>	Disables regular polling for indexing. We recommend that you disable polling for specific repositories in their configuration options, instead.
<code>--Xenable-git-content-hash-resolving-heuristic</code>	This was used to speed up indexing by avoiding an exhaustive lookup of file revisions that give a particular content hash in Git repos. Git manifest support in FishEye removed the need to use this option. It does still apply if a manifest upgrade has not completed.
<code>--Xdisable-head-detection</code>	Disables branch head detection in DVCS systems (Git and Mercurial).
<code>--Xdisable-tags</code>	Disables tag processing in DVCS systems.
<code>--Xlenient-patches</code>	FishEye will attempt to process non-standard patches.

stop

The stop command stops a running FishEye instance.

Options:

<code>--config path</code>	Load configuration from the file at path. Default is \$FISHEYE_INST/config.xml.

fullscan

Usage:

```
fisheyectl fullscan [options] [repname ...]
```

The `fullscan` command requests a full scan of the given repositories, or all repositories if no repository name is given.

Note: The `fullscan` command is only supported for CVS repositories.

Options:

<code>--config path</code>	Load configuration from the file at path. Default is <code>\$FISHEYE_INST/config.xml</code> .

rescan

Usage:

```
fisheyectl rescan [options] repname start end
```

Requests a rescan of the given repository between two specified revision ids.

Note: this operation is not supported by CVS repositories.

Options:

<code>--config path</code>	Load configuration from the file at path. Default is <code>\$FISHEYE_INST/config.xml</code> .

reindex

Usage:

```
fisheyectl reindex [options] [repname ...]
```

Requests a full reindex of the given repositories, or all repositories if no repository name is given.

Options:

<code>--config path</code>	Load configuration from the file at path. Default is <code>\$FISHEYE_INST/config.xml</code> .

scannow

The `scannow` command requests an incremental scan from the command line.

Usage:

```
fisheyectl scannow -s [repname ...]
```

Options:

<code>-s</code>	Run synchronously; i.e. the command will not terminate until the requested indexing operation is complete.

- You can set the poll period to **'Never'** in which case the repo will be viewable from the web UI but the repository will not be polled.
- Repositories may be polled in the demand using this code:

```
fisheyectl.sh scannow [repname ...]
```

This will perform a one-time indexing operation.

Note: `scannow` can also be used on a repository which has a poll period.

backup / restore

FishEye data can be backed up from the admin interface or command line. This page contains the command syntax, options and the required procedure to back up and restore your FishEye instance: [Backing up and restoring FishEye data](#)

svnrules

The svnrules tool ships with FishEye 3.0, and later versions. It parses the SVN log and compares that against the SVN symbolic rules configured in FishEye to find commits that are likely to be branch or tag creation changes for which there are no SVN symbolic rule defined. Find the details here: [Verifying SVN symbolic rules](#)

Read more

- [JVM system properties](#)

Configuring web hooks

Web Hooks allow you to post changeset information to a specific URL. For each commit indexed by FishEye, the changeset data will be sent in JSON format to all configured URLs using a POST request.

You can use this feature to build integration with applications such as issue trackers, continuous integration solutions or notification systems.

- [Adding web hooks](#)
- [Disabling web hooks](#)
- [Demonstration of web hooks](#)
- [Web hook data](#)
- [Sample web hook scripts](#)
 - [Python](#)
 - [Ruby and Sinatra](#)

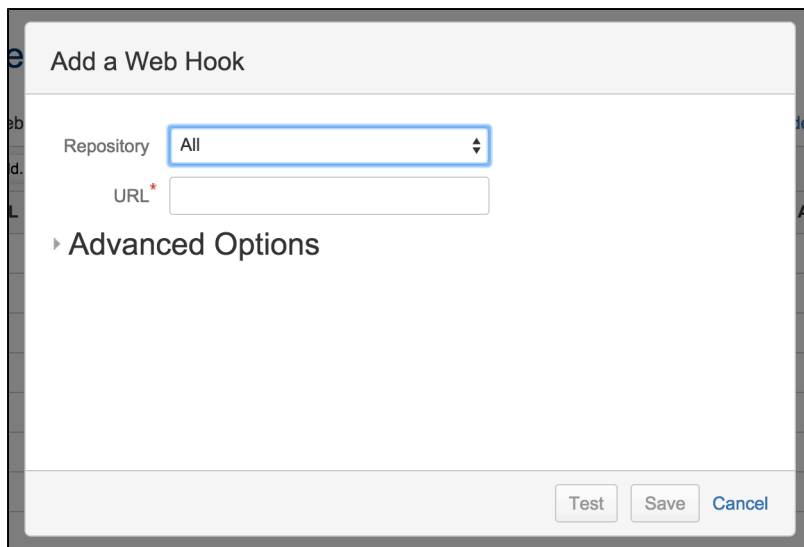
Be aware that Web Hooks can expose private commit information. For example, using a public service could expose your code. Any user with administrative privileges in FishEye can create a Web Hook for any repository.

Adding web hooks

Web Hooks can be added or edited from the configuration page accessible from the Administration interface.

Go to the Admin area and click **Web Hooks**, under 'Repository Settings'.

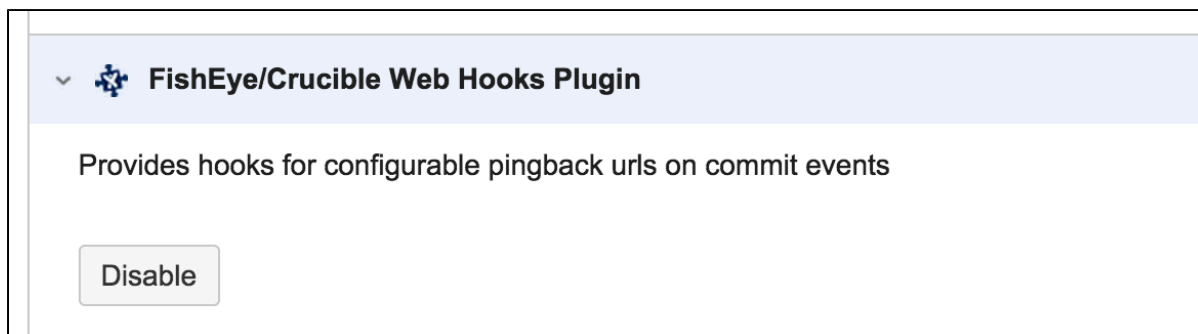
Click **Add** to create a new Web Hook. Choose a repository and enter your URL. Click **Test** and **Save** to verify and store your configuration. Web Hooks also support basic authentication, which can be configured in the **Advanced Options**:



Disabling web hooks

To disable Web Hooks, go to the Administration interface > Plugins

Show **System Plugins** and navigate to the **FishEye/Crucible Web Hooks Plugin** and click **Disable**.



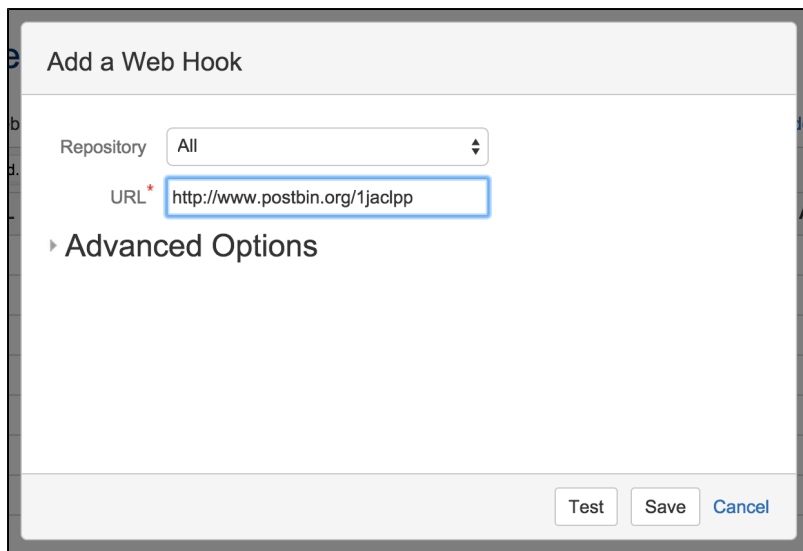
Demonstration of web hooks

[PostBin](http://www.postbin.org) is a service for debugging web hooks.

This is a public service and we do not recommend testing private repositories with public services as your code can be exposed.

Start by going to <http://www.postbin.org> and click **Make a PostBin**.

Grab your PostBin URL and create a Web Hook with it using a test repository.



Add a Web Hook

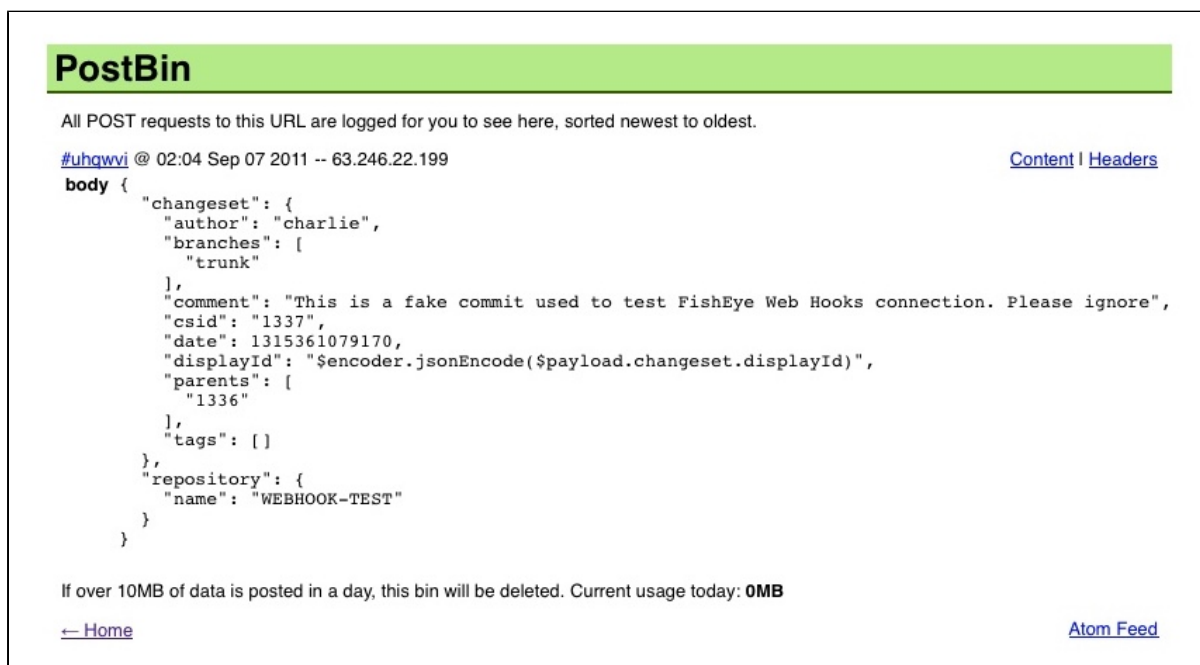
Repository: All

URL: http://www.postbin.org/1jaclpp

Advanced Options

Test Save Cancel

When you click **Test** a dummy changeset will be sent to the PostBin URL. Refresh your PostBin page and you should see the data sent by FishEye.



PostBin

All POST requests to this URL are logged for you to see here, sorted newest to oldest.

[#uhqwvi](#) @ 02:04 Sep 07 2011 -- 63.246.22.199 [Content | Headers](#)

```
body {
  "changeset": {
    "author": "charlie",
    "branches": [
      "trunk"
    ],
    "comment": "This is a fake commit used to test FishEye Web Hooks connection. Please ignore",
    "csid": "1337",
    "date": 1315361079170,
    "displayId": "$encoder.jsonEncode($payload.changeset.displayId)",
    "parents": [
      "1336"
    ],
    "tags": []
  },
  "repository": {
    "name": "WEBHOOK-TEST"
  }
}
```

If over 10MB of data is posted in a day, this bin will be deleted. Current usage today: **0MB**

[← Home](#) [Atom Feed](#)

This is a demonstration of the data sent by FishEye with a configured WebHook. The next step would be to write a service which captures this data and processes it.

Web hook data

Web Hooks will send changeset data in the following JSON representation:

```
{
  "repository": {
    "name" : <repository name>
  },
  "changeset" : {
    "csid" : <changeset id>,
    "displayId" : <changeset id>,
    "author" : <committer>,
    "comment" : <commit message>,
    "date" : <commit date>,
    "branches" : [
      <list of commit branches>
    ],
    "tags" : [
      <list of commit tags>
    ],
    "parents" : [
      <list of parents>
    ]
  }
}
```

Sample web hook scripts

The following script examples show you how to set up Web Hooks in various languages.

Python

This example requires the [simplejson](#) library.

```
import simplejson as json
import pprint
from BaseHTTPServer import BaseHTTPRequestHandler, HTTPServer

class postRequestHandler(BaseHTTPRequestHandler):
    def do_POST(self):
        content_length =
self.headers.getheader('content-length')
        if content_length:
            content_length = int(content_length)
            if self.path.endswith('webhook'):
                self.send_response(200)
                self.end_headers()
                body = self.rfile.read(content_length)
                post_data = json.loads(body)
                print 'Commit added with comment: %s' %
post_data["changeset"]["comment"]

            else:
                self.send_error(404, 'File Not Found %s' %
self.path)

        return

    def run():
        server = HTTPServer(('', 8081), postRequestHandler)
        print 'Server started'
        server.serve_forever()

if __name__ == '__main__':
    run()
```

Ruby and Sinatra

The example below will display the changeset information in the logs of the [Sinatra](#) server.

```
require 'rubygems'
require 'sinatra'
require 'json'

post '/' do
  changeset = JSON.parse(request.body.read)
  puts changeset.inspect
end
```

Enabling Smart Commits

Smart Commits allow repository committers to perform actions such as transitioning JIRA Software issues or creating Crucible code reviews by embedding specific commands into their commit messages.

Read more about [how to use Smart Commits](#).

Because Git and Mercurial allow commits to be attributed to users other than those pushing changes to the repository, if your FishEye instance has any DVCS repositories then Smart Commits will be disabled. This is a security measure to prevent privilege escalation. This only occurs when the feature is first initialized, and will not be disabled if new Git or Mercurial repositories are added afterwards.

Requirements

To get Smart Commits working, an admin on both JIRA Software and FishEye needs to do the following:

- An application link must be configured between FishEye/Crucible and JIRA. See [Adding an Application Link](#).
- Smart Commits must be enabled, as [described below](#).

If the Application Link is configured as OAuth

If the Application Link to JIRA is configured to use OAuth, the committing user must authenticate with JIRA before any Smart Commit will work with JIRA. You can authenticate as follows:

▼ [Click here to expand...](#)

1. Create a test review in FishEye.
2. Log in with the committing user.
3. Open the review and click in "Edit Details".
4. Fill the field "Linked Issue" with the key of some JIRA issue and click in "Link". You'll be prompted to authenticate.
5. Do this for every committing user (no need to create a new review, only link the review to any JIRA issue).

On this page:

- [Requirements](#)
- [Enable Smart Commits](#)
- [Disable Smart Commits](#)
- [FAQ](#)

Related pages:

- [Configuring web hooks](#)
- [Transitioning JIRA issues](#)
- [Write your own Smart Commit](#)

Enable Smart Commits

To enable Smart Commits, open the FishEye **Administration** interface and go to the **Smart Commits** configuration page.

Smart Commits

Smart Commits allow repository committers to perform actions like transitioning JIRA issues or creating Crucible code reviews by embedding specific commands into commit messages.

Put into a commit message, the example below would *close* the JIRA issue BUG-123.

```
BUG-123 #close Fixing a bug
```



Warning

When using Git or Mercurial, it is possible to commit under any username. Ensure you can fully trust your committers when using those systems.

Enable smart commits

JIRA Issue Transitions ☒

Review Creation ☒

Apply

Select the Smart Commits that you would like to enable.

Disable Smart Commits

To disable Smart Commits, deselect the checkboxes for all types of Smart Commit.

FAQ

Q. The Smart Commits administration page is reporting an error, saying "No plugins installed"

A. It is possible that the Smart Commit plugins are disabled in the **Plugins** section of your FishEye instance. Go to **System Settings > Plugins** and open **System Plugins**. Please ensure that the *FishEye/Crucible Review Smart Commits Plugin* and *FishEye/Crucible JIRA Smart Commits Hook Plugin* are enabled (as well as all of their submodules). If you are unable to enable them, please view your FishEye logs for clues and raise a support request at <http://support.atlassian.com/> with your logs attached.

Configuring commit hooks

The **incremental indexing** process causes FishEye to poll all repositories at the specified interval to check for new commits, even though there might not be any new information to index. If you have a large number of repositories (> 100), this can lead to:

- A time lag between a commit being made and it appearing in FishEye.
- A high load on the FishEye server, and on the SCM.

Commit hooks allow you to set up your SCM so that indexing of a repository is triggered by a commit to the repository itself. This means that FishEye only runs the indexing process when necessary, and allows automatic **polling** to be disabled. Commits will appear sooner in FishEye, and the server load will be reduced.

To set up commit hooks you:

1. [Set the REST API token](#) in FishEye/Crucible.
2. Integrate the commit hook with your SCM.

Note that if you add a [Bitbucket Server](#) Git repository to FishEye, a push to the repository will automatically trigger FishEye to run an incremental index, without the need for polling or web hooks – see [Integrating FishEye with Bitbucket Server](#).

On this page:

- [Triggering scanning remotely](#)
- [Integrating with your SCM](#)

- [Bitbucket Cloud and GitHub](#)
- [CVS](#)
- [SVN](#)
- [Perforce](#)
- [Git](#)
- [Mercurial](#)
- [Decreasing polling frequency](#)

Triggering scanning remotely

Once you've set your REST API token you can use it to trigger FishEye scanning when your repository is updated.

The basic way to do this is set up a shell script similar to:

```
echo Triggerring FishEye scan
/usr/bin/curl -X PUT -H "X-API-Key: <YOUR-API-KEY-HERE>" -H
"Content-Type: application/json" <URL>:<PORT></optional
CONTEXT>/rest-service-fecru/admin/repositories/<REPOSITORY-NAME>/increme
ntal-index
```

For example:

```
echo Triggerring FishEye scan
/usr/bin/curl -X PUT -H "X-API-Key: abcdefg123456" -H "Content-Type:
application/json"
http://atlas:8060/fecru/rest-service-fecru/admin/repositories/widget/inc
remental-index
```

Try running the script; if everything is fine, it will output "[]", and will trigger scanning in FishEye. If there are problems, curl will show an appropriate message.

If you're running on Windows, you'll need curl or a similar program. You can download the Windows version of curl [here](#). You'll need to save the script as a batch file (with the .bat extension).

Note: be sure to specify the full path to the curl binary on your system.

Integrating with your SCM

Bitbucket Cloud and GitHub

Both of these hosting services provide service hooks that can be used to trigger repository indexing in FishEye.

Bitbucket

In Bitbucket, go to the admin page for your repository, click **Hooks** and choose **FishEye**.

See the [Bitbucket documentation](#) for more information about setting up a Bitbucket service hook.

GitHub

In GitHub, go to the admin page for your repository, click **Service Hooks** and choose **FishEye** from the available hooks.

CVS

1. Checkout the CVSROOT module of your cvs repository:


```
cvs co CVSR00T
```

2. Edit the CVSR00T/logininfo file.
3. Add the following line to the file:

```
ALL /usr/bin/curl -X PUT -H "X-API-Key: <YOUR-API-KEY-HERE>" -H  
"Content-Type: application/json" -m 20 <URL>:<PORT></optional  
CONTEXT>/rest-service-fecru/admin/repositories/<CVS-REPOSITORY-NAME  
>/incremental-index > /dev/null 2>&1 &
```

e.g.

```
ALL /usr/bin/curl -X PUT -H "X-API-Key: abcdefg123456" -H  
"Content-Type: application/json"  
http://atlas:8060/fecru/rest-service-fecru/admin/repositories/cvs_w  
idget/incremental-index > /dev/null 2>&1 &
```

4. Commit your changes:

```
cvs commit CVSR00T/logininfo
```

SVN

1. Log into your svn server, go to the repository directory, find the hooks subdirectory there:

```
cd /var/www/svn/hooks
```

2. If it doesn't exist, create a new file called post-commit (or post-commit.bat on Windows), make sure it's executable by the user that the svn process runs as:

```
touch ./post-commit  
chmod 755 ./post-commit
```

3. Make sure the file starts with the following shebang line, pointing to your shell:

```
#!/bin/sh
```

4. Add the following to the post-commit file:

```
/usr/bin/curl -X PUT -H "X-API-Key: <YOUR-API-KEY-HERE>" -H  
"Content-Type: application/json" -m 20 <URL>:<PORT></optional  
CONTEXT>/rest-service-fecru/admin/repositories/<SVN-REPOSITORY-NAME  
>/incremental-index > /dev/null 2>&1 &
```

e.g.

```
/usr/bin/curl -X PUT -H "X-API-Key: abcdefg123456" -H  
"Content-Type: application/json"  
http://atlas:8060/fecru/rest-service-fecru/admin/repositories/svn_w  
idget/incremental-index > /dev/null 2>&1 &
```

You can find more details about svn hooks [here](#).

Perforce

1. As a Perforce administrator execute the following command:

```
p4 triggers
```

2. The trigger table form will be presented.
3. Add a field value for the field 'Triggers', named trigger-X, where X is the next number available for the trigger:

```
trigger-04 change-commit //... "/usr/bin/curl -s -o /dev/null -X  
PUT -H X-API-Key:<YOUR-API-KEY-HERE> -H "Content-Type:  
application/json" -m 20 <URL>:<PORT></optional  
CONTEXT>/rest-service-fecru/admin/repositories/<PERFORCE-REPOSITORY  
-NAME>/incremental-index
```

e.g.

```
trigger-04 change-commit //... "/usr/bin/curl -s -o /dev/null -X  
PUT -H "X-API-Key: abcdefg123456" -H "Content-Type:  
application/json"  
http://atlas:8060/fecru/rest-service-fecru/admin/repositories/perfo  
rce_widget/incremental-index
```

4. You can customize the trigger to run only for a specific depot or directory, by replacing the `//...` above (which causes the trigger to be executed for every file) by a standard Perforce file pattern syntax.

You can find more details about Perforce triggers in the [Perforce System Administrator's guide](#).

Git

If you add a Bitbucket Server Git repository to FishEye, a push to the repository automatically triggers FishEye to run an incremental index. You don't have to configure polling to detect new commits, or set up dedicated FishEye web hooks in your Bitbucket Server instance – see [Integrating FishEye with Bitbucket Server](#).

However, if you are creating a hook in Bitbucket Server, see [How to create a simple hook in Bitbucket Server](#).

Otherwise, for Git repositories hosted in other SCMs, follow these steps:

1. Choose the repository you want to trigger the scans from. Usually this is the repository that all of your developers push to, and that you run CI from. Note that hooks are not propagated when cloning repositories.
2. Go to the hooks subdirectory of your repository:

```
cd /var/www/git/project/hooks
```

3. If it doesn't exist, create a new file called `post-receive`. Make sure it's executable by the Git server process.

```
touch ./post-receive
chmod 755 ./post-receive
```

4. Make sure the file starts with the following line, pointing to your shell:

```
#!/bin/sh
```

5. Add the following to the `post-receive` file:

```
/usr/bin/curl -X PUT -H "X-API-Key: <YOUR-API-KEY-HERE>" -H
"Content-Type: application/json" -m 20 <URL>:<PORT></optional
CONTEXT>/rest-service-fecru/admin/repositories/<GIT-REPOSITORY-NAME
>/incremental-index > /dev/null 2>&1 &
```

e.g.

```
/usr/bin/curl -X PUT -H "X-API-Key: abcdefg123456" -H
"Content-Type: application/json"
http://atlas:8060/fecru/rest-service-fecru/admin/repositories/git_w
idget/incremental-index > /dev/null 2>&1 &
```

NOTE: Not all methods of serving a Git repository support commit hooks - if serving over http, you need to use smart-http (either using [git-httpd-backend](#) or a dedicated repository manager like Bitbucket Server). You can find more information about smart http [here](#). Serving the repository over ssh or git-daemon should allow you to run commit hooks as well.

Mercurial

1. Choose the repository you want to trigger the scans from. Usually this is the repository that all of your developers push to, and that you run CI from. Note that hooks are not propagated when cloning repositories.
2. Go to the `.hg` subdirectory of your repository:

```
cd /var/www/hg/project/.hg
```

3. If it doesn't exist create a file named `hgrc`:

```
touch ./hgrc
```

4. Add the following to the `hgrc` file:

```
[hooks]
changegroup = /usr/bin/curl -X PUT -H "X-API-Key:
<YOUR-API-KEY-HERE>" -H "Content-Type: application/json" -m 20
<URL>:<PORT></optional
CONTEXT>/rest-service-fecru/admin/repositories/<HG-REPOSITORY-NAME>
/incremental-index > /dev/null 2>&1 &
```

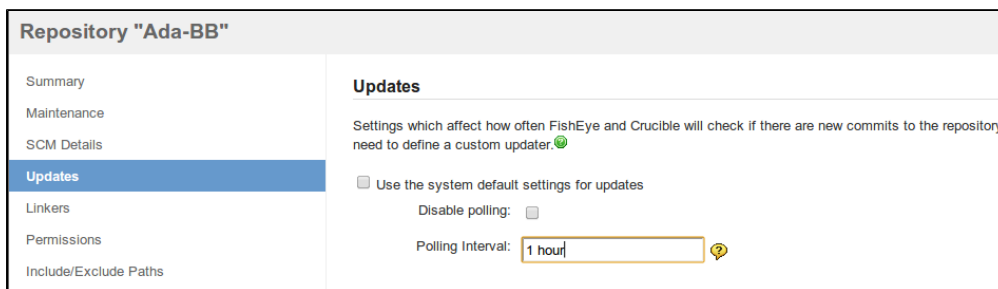
e.g.

```
[hooks]
changegroup = /usr/bin/curl -X PUT -H "X-API-Key: abcdefg123456" -H
"Content-Type: application/json"
http://atlas:8060/fecru/rest-service-fecru/admin/repositories/hg_wi
dget/incremental-index > /dev/null 2>&1 &
```

NOTE: Not all methods of serving a Mercurial repository support commit hooks - if serving over http, you can't use static-http serving.

Decreasing polling frequency

Once your commit hook is set up and successfully notifying FishEye about new commits to your repository, you can decrease the polling frequency on your repository (for example to 1 or 2 hours, instead of the default 1 minute).



The screenshot shows the 'Repository "Ada-BB"' configuration page in FishEye. The left sidebar contains links for Summary, Maintenance, SCM Details, Updates (selected), Linkers, Permissions, and Include/Exclude Paths. The main content area is titled 'Updates' and contains the text: 'Settings which affect how often FishEye and Crucible will check if there are new commits to the repository. need to define a custom updater. 🟢'. Below this, there is a checkbox 'Use the system default settings for updates' which is unchecked. Underneath, there is a 'Disable polling:' checkbox which is checked. Finally, there is a 'Polling Interval:' label followed by a text input field containing '1 hour' and a help icon (question mark).

With commit hooks configured, scheduled polling is only useful if the hook fails, for example because of connectivity issues to the server hosting FishEye. This will decrease the server load, but allow FishEye to still occasionally check for changes, and update the repository if needed.

Setting the REST API token

A REST API token is an alternative way to authenticate to use a REST service in FishEye and Crucible. Instead of specifying a username and password of an authorized user, you can add a request header to the call that includes the REST API token:

```
X-API-Key: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

This is currently only supported for a limited subset of REST services – see <https://docs.atlassian.com/fisheye-crucible/latest/wadl/fecru.html#d2e302>. It allows you to use services that normally require administrator credentials, without exposing those or keeping them (for example in a script or a commit hook). The token cannot be used to log in to FishEye/Crucible or to perform any operations other than the designated REST service calls.

To generate a REST API token, log in to the administrator area of FishEye/Crucible, go to **Authentication** (under 'Security Settings'), then click **Generate new token** in the REST API Token section. A REST API token will be generated and displayed on screen.

You can always prevent access using a previously generated token by deleting it, or generating a new one. Note that this will prevent any existing scripts using the token from working, until you update them with the new token.

See also [Authenticating REST requests](#).

REST API Token [?]

REST API Token `e51c7a12-0845-5700-8000-000000000000`

This token is used to authenticate specific REST calls that otherwise need administrator credentials

Generated On 2012-12-03 15:18 +0000

[Delete token](#)

Configuring ViewVC compatibility

For backwards-compatibility and legacy system support, FishEye maintains a URL-compatibility mode with the ViewVC (formerly known as ViewCVS) and CVSWeb tools. This allows FishEye to replace or extend ViewVC, making use of the URLs set up for ViewVC.

FishEye administrators can configure ViewVC compatibility by clicking **ViewCVS URL Mappings** (under 'Global Settings') in the admin area.

For example, a ViewVC URL of the form

```
http://host/viewcvs.cgi/x/y/z
```

can be viewed in FishEye at

```
http://fisheyehost/viewcvs/x/y/z
```

FishEye can be configured to determine exactly how it provides this compatibility mode. In particular, you can configure how to map ViewVC repository names (cvsroot or root in the query parameter) to FishEye repository names.

You can configure the **Default Mapping** to specify the repository to use if none is supplied in the URL. If a repository name is given in the URL, you can specify how FishEye should translate that to the name of a FishEye repository. Otherwise, FishEye will attempt to use the repository name given in the URL directly.

After editing mappings you will need to restart FishEye.

ViewCVS URL Mappings [?]

Default Mapping: applinks. [Edit](#)

Viewcvs Name	Repository	Operations
test	activeobjects	Edit Delete

Customizing the front page

This feature is not fully functional in FishEye 2.2 due to a [bug](#). To read more information about this, please see the [issue](#) in our public issue tracker.

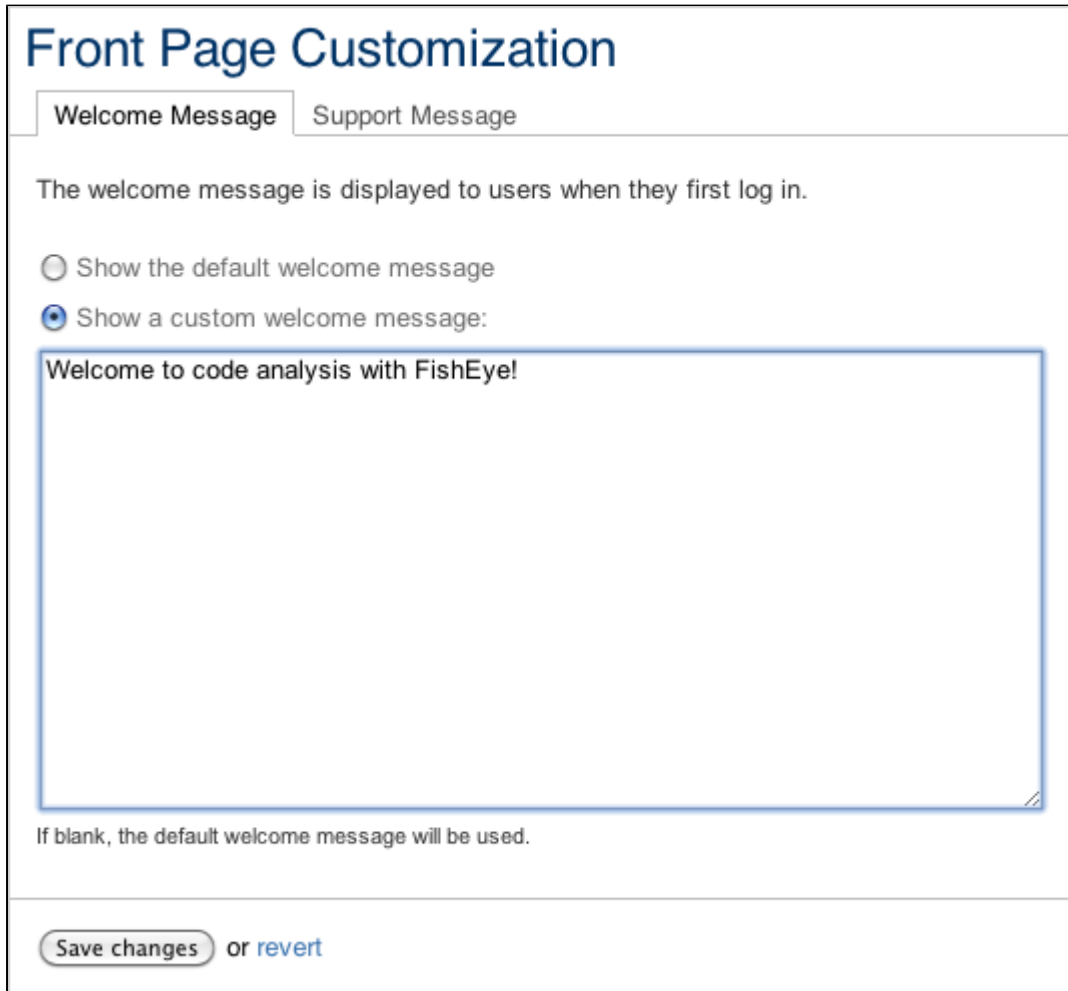
You can customize the welcome message and the support contact information for your organization that users see when they first log in.

To customize these messages, go to the [admin area](#), and click **Front Page Customization** (under 'Global Settings' in the left navigation bar).

Click either of the **Welcome Message** or **Support Message** tabs, and then **Show a custom... message**.

Enter your message text into the box.

Click **Save changes**. Your changes are applied immediately.



The screenshot shows the 'Front Page Customization' interface. At the top, there are two tabs: 'Welcome Message' (selected) and 'Support Message'. Below the tabs, a text box contains the message 'Welcome to code analysis with FishEye!'. Above this text box, there are two radio buttons: 'Show the default welcome message' (unselected) and 'Show a custom welcome message:' (selected). Below the text box, a note states 'If blank, the default welcome message will be used.' At the bottom, there are two buttons: 'Save changes' and 'or revert'.

Restoring the default messages

To revert to the default Welcome or Support messages, simply delete the text shown in the text box and click **Save changes**.

Manually editing the opening screen

You can directly edit the XML in the `config.xml`, located in your `<FishEye install directory>`.

Simply add the following XML tags to the `config.xml` file:

```
<content>
  <front-page-message>Example welcome message here</front-page-message>
  <support-message>Example support message here</support-message>
</content>
```

Using HTML

The content in the welcome screen can be arranged using tables, images or HTML tags such as the following:

```
<a href="http://www.atlassian.com">Link to Atlassian Home Page</a>
```

Configuring raw file download settings

This page describes how to configure FishEye to either immediately display downloaded files in the browser, or show the browser's "Save As" dialog.

Configuration

As a FishEye administrator:

1. Log in to the admin area and click **Server** (under 'Global Settings').
2. Click **Edit Details** under 'File Download Setting' and choose a setting:

Mode	Characteristics
Smart (default)	Files that can safely be displayed inline in the browser, will be. Files that are on FishEye's default list of file extensions that are considered dangerous, will always be accessed via the "Save As" dialog. For files that are considered dangerous, but consist of only plain text (such as .html files), this mode will attempt to force the browser to show (but not execute) the file.
Secure	All files, regardless of whether they are considered dangerous or not, are forced to be always accessed via the "Save As" dialog instead of being displayed inline.
Insecure	All files downloaded via FishEye will be displayed in the browser after clicking.

3. Click **Update**.

 Internet Explorer versions 8 and 9 will not display safe content as plain text.

Default file extensions that FishEye will consider as executable in the browser

In Smart mode, downloading files with any of the following extensions will always trigger the "Save As" dialog, instead of displaying the contents of the file in the browser directly, except in FireFox, Safari and Chrome.

File Extensions	Comments
.txt	Normally these are just text files, but a malicious user could write html in this file, and cause XSS inject attacks if the browser executes the content instead of displaying it. Can be made to be safely displayed in all supported browsers, except Internet Explorer.
.htm , .html, .xhtml, .xml, .svg	Similar to .txt, these are text files, but contains html. Can be made to be safely displayed in all supported browsers, except Internet Explorer.
.swf, .flv , .f4v, .f4p, .f4b	Flash object file.
.cab	Windows Cabinet File.

To add items to the default list in FishEye, please contact Atlassian Support.

Software update notifications

FishEye (and Crucible) can detect and notify you of new versions of the software.

In the Admin screen, you can set a background thread to poll Atlassian's servers and report when a new version of FishEye or Crucible has been released.

To use the Update Notification:

1. In the FishEye [admin area](#), click **Server** under 'Global Settings'.
2. Under 'Update Notifications', click **Enable** to switch on regular polling of the Atlassian servers.
3. Once enabled, click **Edit config** to adjust the polling period.

 A status of 'Unknown' is displayed when polling has not yet been enabled.

Enabling Access Logging in FishEye

To enable access logging in FishEye 3.0 and later:

1. Stop Fisheye/Crucible,
2. Create the file `<FishEye install directory>/content/WEB-INF/jetty-web.xml` with the following content:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Configure PUBLIC "-//Mort Bay Consulting//DTD
Configure//EN" "http://www.eclipse.org/jetty/configure.dtd" >
<Configure class="org.eclipse.jetty.webapp.WebAppContext">
  <Set name="handler">
    <New
class="com.cenqua.fisheye.web.jetty.FishEyeRequestLogHandler">
      <Set name="requestLog">
        <New id="RequestLogImpl"
class="org.eclipse.jetty.server.NCSARequestLog">
          <Arg><SystemProperty name="jetty.logs"
default="/var/log/" />fisheye-access-yyyy_mm_dd.log</Arg>
          <Set name="retainDays">90</Set>
          <Set name="append">true</Set>
          <Set name="extended">false</Set>
          <Set name="LogTimeZone">GMT</Set>
        </New>
      </Set>
    </New>
  </Set>
</Configure>
```

3. Restart Fisheye/Crucible.

This will create an access log in `<FishEye install directory>/var/log/fisheye-access-yyyy_mm_dd.log` format (e.g. `fisheye-access-2010_03_17.log`). If you want to change the path to your FISHEYE_INST directory, change the `default="/var/log/"` to the path to the log folder in FISHEYE_INST.

The log directory must exist

If the path to the log directory given by the `default` attribute of the `SystemProperty` tag (defined in the line 10 in the `jetty-web.xml` above) does not exist, then FishEye will fail to start and will not log any error message.

The path given in the example below is correct when FISHEYE_INST and the `<FishEye install directory>` are the same directory, otherwise please use the absolute path of your FISHEYE_INST/v

ar/log directory.

Log format

The logs are written in [NCSA format](#):

```
172.20.5.186 - - [17/Mar/2010:22:50:21 +0000] "GET
/fe/commitSparkline.do?w=280&h=48&context=repository&repname=TestRepo&ou
tputtype=image HTTP/1.1" 200 256
172.20.5.186 - - [17/Mar/2010:22:50:21 +0000] "GET
/fe/commitSparkline.do?w=280&h=48&context=repository&repname=npanday&out
puttype=image HTTP/1.1" 200 177
172.20.5.186 - - [17/Mar/2010:22:50:21 +0000] "GET
/fe/commitSparkline.do?w=280&h=48&context=repository&repname=jutils&outp
uttype=image HTTP/1.1" 200 775
172.20.5.186 - - [17/Mar/2010:22:50:21 +0000] "GET
/fe/commitSparkline.do?w=280&h=48&context=repository&repname=P4&outputty
pe=image HTTP/1.1" 200 177
172.20.5.186 - - [17/Mar/2010:22:50:21 +0000] "GET
/fe/commitSparkline.do?w=280&h=48&context=repository&repname=Rails&outpu
ttype=image HTTP/1.1" 200 1311
172.20.5.186 - - [17/Mar/2010:22:50:22 +0000] "GET
/fe/commitSparkline.do?w=280&h=48&context=repository&repname=FE-2363&out
puttype=image HTTP/1.1" 200 128
```

Please refer to the [Jetty documentation](#) for more configuration options.

Compatibility

If you are using an earlier version of FishEye than FishEye 2.7.8, replace `com.cenqua.fisheye.web.jetty.FishEyeRequestLogHandler` by `org.mortbay.jetty.handler.RequestLogHandler`.

FishEyeRequestLogHandler was added in 2.7.8 to fix an issue where the user credentials would not be added to the NCSA log: [FE-3040](#).

FishEye and Crucible Cookies

- [Authentication Cookies](#)
- [The 'Remember Me' Cookie](#)
 - [Cookie Key and Value](#)
 - [Use of Cookie for Authentication](#)
 - [Life of 'Remember Me' Cookies](#)
- [Other cookie usage](#)
 - [crucibleprefs1](#)

Authentication Cookies

FishEye/Crucible uses two cookies:

- The FSESSIONID cookie is created by the Jetty application server and used for session tracking purposes.
- The 'remember me' cookie, `remember`, is generated by FishEye when the user selects the **Remember me** checkbox on the login page. This option is only available in a FishEye only instance. A Crucible instance will always have this option turned on.

 You can read about cookies on the [Wikipedia page](#).

The 'Remember Me' Cookie

The 'remember me' cookie is a long-lived HTTP cookie. This cookie can be used to authenticate an unauthenticated session. FishEye/Crucible generates this cookie when the user logs in and in FishEye only mode also has the **Remember me** checkbox on the login page.

Cookie Key and Value

The cookie key is `remember` and is not configurable.

The cookie contains the user's username in plain text, a unique identifier plus a securely-generated random string.

Use of Cookie for Authentication

When a user requests a web page, if the request is not already authenticated via session-based authentication or otherwise, FishEye/Crucible will match the 'remember me' cookie (if present) against the token stored for the user in the FishEye/Crucible database (if present).

If the random string matches the value stored in the database and the cookie has not expired, the user is authenticated.

Life of 'Remember Me' Cookies

The 'remember me' cookie expires after 365 days of inactivity.

Other cookie usage

`crucibleprefs1`

`crucibleprefs1` is a single cookie, which is used to store different user preferences. These are used to make actions the user has performed in the UI, (e.g setting the syntax used for a snippet) 'sticky'.

This saves the user having to continually set these values each time they visit FishEye/Crucible.

The complete list of values stored in `crucibleprefs1` are:

No.	Full Name	Cookie Name	Default Value	Possible Values
1	longTagList	ltl	'M'	' L , M , H '
2	showFileHistImage	sfhi	'Y'	OPEN*
3	showAuthorChart	sac	'Y'	OPEN*
4	showQuickRecent	sqr	'Y'	OPEN*
5	annotateMode	annm	'age'	OPEN*
6	dirListOrder	dlo	'p'	' f ', p , l '
7	csShowDiffs	csSD	'false'	' false ', true '
8	csTruncateDiffs	csTD	'true'	' true ', false '
9	hideHiddenDirectories	hhd	'Y'	' N ', Y '
10	hideEmptyDirectories	hed	'N'	' N ', Y '
11	includeUserOnHome	iuh	'Y'	' N ', Y '
12	hideDeletedFiles	hdf	'N'	' N ', Y '
13	alwaysExpandChangesets	aec	'N'	' N ', Y '

14	viewMode	hvm	'p'	' p.l '
15	diffColour	dc	'2'	' 2.1 '
16	diffMode	dm	'u'	' u.y '
17	diffWrap	dw	'h'	' s.h '
18	diffContextBar	dcbt	's'	' s.h '
19	selectedPanel	slp	'a'	' t.a.o '
20	showPanel	shp	'Y'	' N.Y '
21	kOpts	k	'kv'	' v.kv.o.k '
22	activityStreamView	asv	'all'	' cru.jira.fe.all '
23	ignoreSlurpWarningDialog	iswd	'N'	' N.Y '
24	showEditModeHelp	semh	'Y'	' N.Y '
25	reviewHistoryTab	rht	'details'	' details.timeline '
26	showAuthorColumn	authcol	'Y'	' N.Y '
27	showRevisionColumn	revcol	'Y'	' N.Y '
28	showLineNumColumn	lnumcol	'Y'	' N.Y '
29	implicitLines	impllines	'all'	' none.all.auth '
30	snippetSyntax	ssyn	'txt'	OPEN* - the syntax highlighting to use for Snippets
31	snippetProject	sprj	"	OPEN* - the name of the project to default to when creati
32	expCsHeader	xcsh	'N'	' N.Y '
33	lastbr	lsbr	"	** <pre>{ repName: { "type": "Branch" or "Tag", "name": branchName } }</pre>
34	visBranchSet	vbs	"	** <pre>{ "repoName1" : { bl: ["branchName1", "branchName2", }, "repoName2" : { bl: ["branchName1", "branchName2", }, "repoName3" : { bl: ["branchName1", "branchName2", } }</pre>

35	visHighlighter	vhl	"	** <pre>{ "repoName1": {"hl": 'Author'}, "repoName2": {"hl": 'Search'}, "repoName3": {"hl": 'Lineage'}, ... }</pre>

NOTES

* OPEN means the value is not constrained by code

** JSON Values: examples are given instead of Possible Values.

Configuring the FishEye web server

To configure the server settings, click **Server** (under 'Server Settings') in the Admin area.

For information on configuring SSL for FishEye, see [FishEye SSL configuration](#).

Some settings on this page require you to restart FishEye for changes to take effect.

If you set proxy scheme/host/port values, then sessions accessing the port locally rather than through the reverse proxy will be redirected to the external proxy URL. This means that Application Links set up using the internal address will fail.

If you want to use a connector both internally and externally, you must **not** set proxy values in the FishEye configuration, but instead ensure that your proxy server is setting the X-Forwarded-Host header in the requests it sends to FishEye. Note that the X-Forwarded-Host header does not contain the URL scheme, so the scheme must be the same internally and externally.

Related pages:

- [Integrating with other web servers](#)
- [FishEye SSL configuration](#)
- [Subversion client settings](#)

Screenshot: Configuring server settings

Server Settings

Web Server

HTTP Bind **j2ee_eac-fisheye.atlassian.com.mgmt:38009**
 SSL Bind **not set**
 SSL Keystore **not set**
 SSL Truststore **not set**
 Web context **crucible**

If you have set up another web server to redirect requests to FishEye/Crucible, set the scheme, host and port of the URL it redirects from here. You must also set a matching Site URL below.

Proxy scheme **https**
 Proxy host **extranet.atlassian.com**
 Proxy port **443**
 Site URL **https://extranet.atlassian.com/crucible**
 AJP13 bind **not set**
 Server timezone **not set (defaulting to US/Central)**

[Edit settings](#)

Setting	HTTP Bind
Description	The hostname the FishEye web server will bind to. This can take the form of a host name and port number, or you can leave the host name blank. If no host name is specified, then FishEye will bind to all available interfaces.
Examples	You can use bindings like the following: Host name and port number: hostname:8060 Port number only(requires a leading colon): :8060 IP address and port number: 10.0.0.11:60 (At least one of 'AJP13 Bind' or 'HTTP Bind' must be set.) Do not add http:// to the front i.e. do not add http://hostname:8060 or http://10.0.0.11:60. Simply define hostname:port or IpAddress:port
Restart required	Yes
Setting	Web context

Description	By default, the FishEye application can be accessed via http://HOST:PORT/ , where HOST and PORT are defined as above. You can force the FishEye application to be hosted on a different 'context' or 'path' by specifying a value here.
Example	If you specify a web context of 'fisheye' then FishEye will be accessible from http://HOST:PORT/fisheye/ instead of http://HOST:PORT/ .
Restart required	Yes

Setting	Proxy scheme
Description	Can be set if you are forwarding through to FishEye from another web server. If you set proxy values, you must also add a matching site URL (see below).
Example	Valid settings are http and https.
Restart required	Yes

Setting	Proxy host
Description	Can be set if you are forwarding through to FishEye from another web server. If you set proxy values, you must also add a matching site URL (see below).
Example	A valid setting would be www.example.com, where 'example' is the domain name of your web server.
Restart required	Yes

Setting	Proxy port
Description	Can be set if you are forwarding through to FishEye from another web server. If you set proxy values, you must also add a matching site URL (see below).
Example	The port number of the web server, an integer from 0 through 32,765.
Restart required	Yes

Setting	AJP13 Bind
Description	Leave this empty unless you are connecting to FishEye/Crucible using the AJP protocol (for example via Apache and mod_jk.) The bind host name for ajpv13. If no host name is specified, then FishEye will bind to all available interfaces.

Examples	You can use bindings like the following: Host name and port number: <pre>hostname:8009</pre> Port number only(requires a leading colon): <pre>:8009</pre> IP address and port number: <pre>10.0.0.11:8009</pre> (At least one of 'AJP13 Bind' or 'HTTP Bind' must be set.) Do not add http:// to the front i.e. do not add http://hostname:8060 or http://10.0.0.11:60. Simply define hostname:port or IpAddress:port
Restart required	Yes

Setting	Server timezone
Description	The time zone to use within FishEye. This time zone is used when displaying dates and parsing EyeQL date expressions. If blank, then the time zone of the server running FishEye is used.
Example	This defaults to the FishEye server's time zone, but you can select another zone from the drop-down list.
Restart required	No

Setting	Site URL
Description	The base URL for this FishEye instance. If not specified, FishEye will attempt to determine this value. You need to set this if you are forwarding through to FishEye from another web server, or if FishEye does not correctly determine the host name you want FishEye users to see FishEye on. This URL is used to: • Generate links in notification e-mails. • Generate the gadget subscription URL, which other applications use to access XML data. • Generate the Display URL during Application Link creation. If you have set a proxy scheme/host/port, the values used there must match the site URL. Do not include a trailing slash ('/') at the end of the URL.

Example	http://www.example.com/apps/fisheye
Restart required	No

FishEye SSL configuration

This page describes how to use SSL with FishEye.

On this page:

1. Obtain an SSL certificate
 - Self-signed certificate
 - Certificate authority certificate
2. Configure FishEye SSL
3. Remove HTTP Access

1. Obtain an SSL certificate

An SSL certificate is required in order for SSL to work in FishEye. There are two ways to obtain one:

- by [signing one yourself](#), or
- by getting one [signed by a Certificate Authority](#).

Self-signed certificate

Self-signed certificates are useful in cases where you require encryption, but do not need to verify the website's identity. They are commonly used for testing and on internal corporate networks (intranets). If a certificate is not signed by a Certification Authority (CA), users may get prompted that the site is untrusted. They may then have to perform several steps to "accept" the certificate before they can access the site. This usually only occurs the first time the site is accessed.

Self-signed certificates are not supported when creating Application Links from an Atlassian OnDemand instance.

Please note, the following approach to create the certificate uses Java's [keytool](#), and has been designed for use with Java 1.6. There are other tools for generating certificates such as [openSSL](#), but the examples will be for keytools, unless the functionality required does not exist in keytools.

To obtain a self-signed certificate:

1. Run the following command to create a new keystore file in the FishEye install directory (if one does not already exist):
 - Windows:

```
"%JAVA_HOME%\bin\keytool" -genkey -keystore
\path\to\keystore.kst -alias fisheye -keyalg RSA
```

- Unix/Linux:

```
$JAVA_HOME/bin/keytool -genkey -keystore /path/to/keystore.kst
-alias fisheye -keyalg RSA
```

2. When the keytool utility prompts you with "What is your first and last name?", enter the fully qualified hostname of the server running FishEye/Crucible. Do not enter your first name and last name.

What is the fully qualified hostname of my server?

The fully qualified hostname of your server is the name you would type in your web browser, after the `http://` section, to access a conventional website. When the client web browser examines the certificate, it checks this field to make sure it matches the hostname. If it does not, the browser may prevent access to the site and will at least generate messages saying that there is a mismatch. An example of a qualified hostname is: `support.atlassian.com`

- When the keytool utility prompts you to enter the keystore password and key password, enter your desired passwords. You must also specify these passwords in the FishEye/Crucible web admin (recommended) or specify these passwords directly in your `config.xml` in the corresponding attributes (see [Configure FishEye/Crucible SSL](#) below).

```
<web-server context="/crucible"
  site-url="http://localhost:6060/crucible/">
<http bind=":6060"/>
<ssl bind=":6443" keystore="/path/to/keystore.kst"
keystore-password="password123"
  truststore="/path/to/keystore.kst"
truststore-password="password123"/>
</web-server>
```

- You now have the minimal requirements to run SSL in FishEye. Next, configure FishEye/Crucible to use SSL as described in the [Configure FishEye/Crucible SSL](#) section below.

Certificate authority certificate

Digital Certificates issued by trusted 3rd party Certification Authorities (CAs) provide verification of the identity of your website. Many CAs simply verify the domain name and issue the certificate. Other CAs such as [VeriSign](#) also verify the existence of your business, the ownership of your domain name and the authority to whom the certificate application was made, thereby providing a higher standard of identification and authenticity.

A list of CA's can be found [here](#). Some of the most well-known CAs are:

- [Verisign](#)
- [Thawte](#)
- [CAcert](#) (relatively new CA, providing free CA certificates)

This list is not an endorsement of the given certificate authorities by Atlassian, and is only provided as an example.

To obtain a certificate signed by a CA:

- Follow the instructions from the certificate authority you want your certificate signed by. Most CAs have their own instructions for you to follow, e.g. [GoDaddy](#) and [VeriSign](#).
- If your CA requires a certificate signing request (CSR), use the following command:
 - Windows:

```
"%JAVA_HOME%\bin\keytool" -certreq -alias fisheye -keystore
\path\to\keystore.kst -file \path\to\fisheye.csr
```

- Unix/Linux:

```
$JAVA_HOME/bin/keytool -certreq -alias fisheye -keystore
/path/to/keystore.kst -file /path/to/fisheye.csr
```

3. Ensure that the keystore being used to generate the signing request contains an existing key/cert keypair. If you don't already have a key/cert keypair, follow the steps in [Self Signed Certificates](#) to generate a keypair.
4. If the certificate you receive from the CA is not in a format that keytool understands then the `openssl` command can be used to convert formats since `keytool` does not perform any of these conversions. For example, to convert between DER and PEM formats:

```
openssl x509 -in certificate.der
            -inform DER -outform PEM -out certificate.crt
```

If you have a key and a certificate in separate files you will need to combine them into a PKCS12 format file for loading into a new keystore. See [these instructions](#) for information on how to do so.

5. Load the certificate into the keystore, as follows:

- CA certificate in PEM format:
 - Windows:

```
"%JAVA_HOME%\bin\keytool" -import -keystore
\path\to\keystore.kst
    -alias fisheye -file \path\to\certificate.crt
    -trustcacerts
```

- Unix/Linux:

```
$JAVA_HOME/bin/keytool -import -keystore
/path/to/keystore.kst
    -alias fisheye -file /path/to/certificate.crt
    -trustcacerts
```

- CA certificate in PKCS12 format:
 - Windows:

```
%JAVA_HOME%\bin\keytool -importkeystore
    -srckeystore \path\to\certificate.pkcs12
    -srcstoretype PKCS12 -destkeystore
\path\to\keystore.kst
```

- Unix/Linux:

```
$JAVA_HOME/bin/keytool -importkeystore
    -srckeystore /path/to/certificate.pkcs12
    -srcstoretype PKCS12 -destkeystore
/path/to/keystore.kst
```

6. You will now have a keystore file with your certificate in it. Next, configure FishEye/Crucible to use SSL as described in the [Configure FishEye/Crucible SSL](#) section below.

If your organization utilizes a wildcard certificate to protect your entire domain you will likely have several files in PKCS#12 format that need to be converted into a Java recognized keystore (JKS). For example, given these four files, use the instructions below to create the keystore:

- `star_yourdomain_com.crt` - wildcard domain certificate
- `star_yourdomain_com.key` - private key
- `DigiCertCA.crt` - intermediate certificate chain

- TrustedRoot.crt - root certificate

1. Create a combined PKCS#12 file:

```
openssl pkcs12 -export
-in <wildcard domain certificate> -inkey <private key>
-certfile <intermediate certificate chain>
-CAfile <root certificate> -caname root -name jetty -out
<p12 output file>
```

For example:

```
openssl pkcs12 -export -in star_yourdomain_com.crt -inkey
star_yourdomain_com.key
-certfile DigiCertCA.crt
-CAfile TrustedRoot.crt -caname root -name jetty -out
fecru.p12
```

2. Create the JKS file from the PKCS#12 file generated above – use the same password for the both deststorepass and destkeypass:

```
keytool -importkeystore
-deststorepass <MY_DESTINATIONSTORE_PASSWORD> -destkeypass
<MY_DESTINATIONKEY_PASSWORD>
-destkeystore <keystore file name> -srckeystore <p12 output
file>
-srcstoretype PKCS12 -srcstorepass
<MY_PKC12_KEYSTORE_PASSWORD> -alias jetty
```

For example:

```
keytool -importkeystore -deststorepass jks_password
-destkeypass jks_password
-destkeystore fecru.jks -srckeystore fecru.p12
-srcstoretype PKCS12 -srcstorepass p12_password -alias jetty
```

2. Configure FishEye SSL

FishEye requires additional configuration in order to use SSL. This configuration can be done from the web admin as described below (recommended), or added directly into your `config.xml`. Setting the bind address and the keystore and password field, as described below, will enable SSL in FishEye. You will need to restart your server after updating the settings, for the changes to take effect.

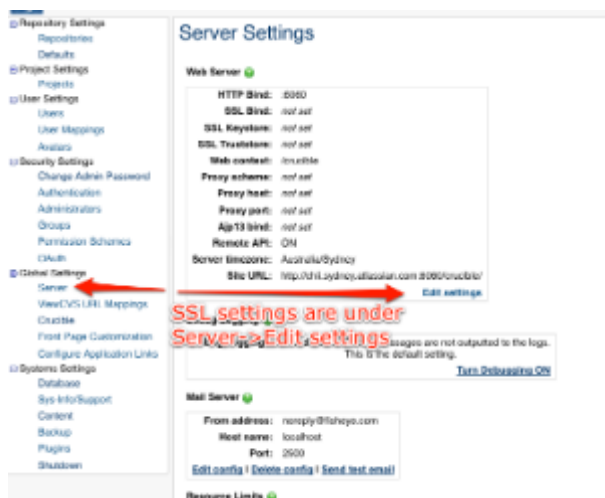
To configure FishEye to use SSL:

1. Navigate to the FishEye admin area and click **Server**, under 'Global Settings' (see 'Server Settings' screenshot below).
2. Click **Edit settings** in the 'Web Server' section.
3. Update the following fields (see 'Example SSL Settings' screenshot below):

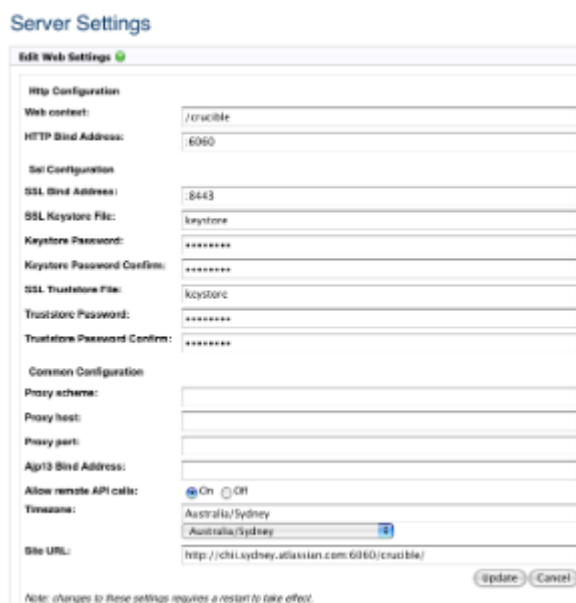
Field	Description	Example value
SSL Bind Address	The SSL port. Most browsers default to 443 as the SSL port for the https protocol. On Linux, only root users can bind to ports below 1024. For security reasons, we recommend running FishEye/Crucible as a dedicated, non-root user. To bind to a port lower than 1024, see How Do I Use Port 80 or 443 on My Server as a Non-Root User on Linux for possible workarounds.	443
SSL Keystore	Path to the keystore file (as generated in the steps above). This path can be a relative path - e.g., putting in <code>keystore.kst</code> will mean that the file is in <code><FishEye install directory>/keystore.kst</code> (or <code>FISHEYE_INST/keystore.kst</code> if it has been set up)	keystore.kst
SSL Keystore Password	The password you used in the above step when creating the keystore file. If you did not set a password, leave this empty. See the passwords section	
SSL Truststore	Path to the truststore file. May be the same as the keystore. Truststore is a list of trusted CAs. Format is the same as the keystore entry.	
SSL Truststore Password	The password you used in the above step when creating the truststore file. If you did not set a password, leave this empty. See the passwords section	

4. Restart FishEye.

Screenshots: FishEye server settings (click to view larger images)



1. Server Settings



2. Example SSL Settings

3. Remove HTTP Access

If you want your FishEye instance to be accessible via HTTPS only, you must remove the `<http bind=":8060"/>` tag from `config.xml`.

Integrating with other web servers

Error rendering macro 'viewport-redirect' : null

Third Party Web Server Support

Please note that support for third-party web servers is provided on a best-effort basis. Please see [Atlassian Support Offerings](#) for more information.

FishEye has a built-in web server, but commonly runs in an environment that has its own web server. You can easily proxy through to FishEye from this primary web server, so that it appears as if FishEye is part of the primary web server.

In most situations, FishEye can determine the host and port of the primary web server automatically. This is useful when you have multiple virtual hosts proxied through to the one FishEye instance.

If it appears FishEye is having trouble automatically detecting the primary web server's host and port, you will need to set the **Proxy host** and **Proxy port** parameters. If the primary web server is running on WEBHOST:80 and FishEye is running on FEHOST:8060, then you can set FishEye's **Proxy host** and **Proxy port** parameters to WEBHOST and 80. See [Configuring the FishEye web server](#) for details on making these changes. The examples below show how the information will be saved to the FishEye configuration file (config.xml):

```
<http bind=":8060" proxy-host="extranet.example.com" proxy-port="80"
proxy-scheme="http"/>
```

If the primary web server is using SSL, add the parameter proxy-scheme="https" and also add the correct port as in the following example:

```
<http bind=":8060" proxy-host="extranet.example.com" proxy-port="443"
proxy-scheme="https"/>
```

Note: FishEye sends cookies in clear text when a user makes a non-SSL request (the cookie is made secure if the incoming request uses SSL). If SSL acceleration is done by primary reverse proxy, it is advisable to rewrite all cookies at the reverse proxy level to make them secure.

You will probably want FishEye to appear in a subdirectory of the primary server. In that case, you need to set FishEye's **web context** parameter (again, see [Configuring the FishEye web server](#)).

```
<web-server context="/fisheye">
<http bind=":8080" proxy-host="extranet.example.com" proxy-port="443"
proxy-scheme="https"/>
</web-server>
```

The rest of the page assumes you have set this value to fisheye.

i You will need to **restart** FishEye before any of the above parameters take effect.

Then, configure your primary web server as described below.

Apache

The easiest way to proxy through to FishEye is using the ProxyPass and ProxyPassReverse directives which require the mod_proxy module. Add this section to your Apache configuration:

```
ProxyPass /fisheye http://FEHOST:8060/fisheye
ProxyPassReverse /fisheye http://FEHOST:8060/fisheye
```

When you proxy to the FishEye host you should set the web context in the [FishEye web server configuration](#) to the same path as the one you are proxying. In the example above the FishEye web context should be set to "fisheye".

If you want Apache to serve FishEye's static content, then you can do something like this instead:

```
<Directory "<FishEye install directory>/content/static" >
    Allow from all
    AllowOverride None
</Directory>
Alias /fisheye/static <FishEye install directory>/content/static
ProxyPass /fisheye/static/ !
ProxyPass /fisheye http://FEHOST:8060/fisheye
```

An alternative to using ProxyPass is to use mod_rewrite with the [P] flag.

AJP

FishEye also supports AJPv13 connectivity. For more information, please see [Using AJP with FishEye](#).

Troubleshooting

Running FishEye as a Windows service

For FishEye 3.4 and later, 32-bit and 64-bit Windows installers are available. Each installer sets up the service wrapper, adds FishEye as a Windows service, and starts the service, automatically.

The installer is the recommended way to install FishEye as a service on Windows. See [Installing FishEye on Windows](#).

If you use the FishEye installer for Windows, you can edit JVM settings using the tool included with the installer. See [Setting JVM system properties](#) for more information.

The content on this page describes the old, deprecated method of configuring FishEye as a Windows service, and is only retained here for informational purposes.

On this page:

- [Running FishEye as a Windows service](#)
- [Installing the Java Service Wrapper](#)
- [Setting FishEye environment variables for Windows Services](#)
- [Troubleshooting](#)
 - [Extracting files from wrapper.zip](#)
 - [Warning when using 64-bit Java JDK](#)
 - [Wrapper configuration and "-server" parameter](#)

Running FishEye as a Windows service

FishEye can be run as a service under Microsoft Windows using a [Java Service Wrapper](#).

The service wrapper provides the following benefits:

- Allows FishEye, which is a Java application, to be run as a Windows Service.
- No need for a user to be logged on to the system at all times, or for a command prompt to be open and running on the desktop to be able to run FishEye.
- The ability to run FishEye in the background as a service, for improved convenience, system performance and security.
- FishEye is launched automatically on system startup and does not require that a user be logged in.
- Users are not able to stop, start, or otherwise tamper with FishEye unless they are an administrator.
- Provides advanced failover, error recovery, and analysis features to make sure that FishEye has the maximum possible uptime.

 Please note that:

- This page should be read in conjunction with [Installing FishEye on Windows](#).
- You should use 32-bit Java to run the service wrapper provided via the link in the install instructions below, even on a 64-bit machine. (64-bit Java will not work with the 32-bit wrapper. There is no 64-bit community "free" version of the service wrapper.)
- You should use the Java JDK, rather than the JRE, to take advantage of the `-server` parameter, provided in the Wrapper configuration of [wrapper.zip](#), which enables the [Java HotSpot\(TM\) Server VM](#). See the [note](#) below for details.

Installing the Java Service Wrapper

To install the Java Service Wrapper on Windows:

1. Download `wrapper.zip` from [here](#).
2. Unzip the wrapper zip file into your `<FishEye install directory>` (that is, the directory into which FishEye was originally installed). Note, the resulting folder structure should be `<FishEye install directory>\wrapper` or `<FishEye install directory>\wrapper\bin`, etc and NOT `<FishEye install directory>\wrapper\wrapper` or `<FishEye install directory>\wrapper\wrapper\bin`. The location of the wrapper directory is important.
3. Tell the wrapper where to find the Java JDK by editing the `<FishEye install directory>\wrapper\conf\wrapper.conf` file, replacing this:

```
# Java Application
wrapper.java.command=java
```

with the following, and comment out the option you don't wish to use:

```
# Java Application

# Option 1: If you have JAVA_HOME defined in your Windows system
environment variables (for example, if JAVA_HOME is defined as
C:/Java/bin, then you can use:
wrapper.java.command=%JAVA_HOME%/java

# Option 2: If you have multiple JDKs installed, and you don't want
to use a Windows environment variable to specify which one to use,
provide the absolute path to where the JDK is installed (e.g.
C:/Java/jdk1.7.0_05/bin/java):
wrapper.java.command=C:/<path to Java location>/bin/java
```

To get confirmation in the wrapper log that the wrapper is using the correct Java JDK, add the following

lines to the wrapper .conf file:

```
# Tell the Wrapper to log the full generated Java command line.
wrapper.java.command.loglevel=INF
```

You can find the logs at <FishEye install directory>\var\log\wrapper.log.

4. Set the FISHEYE_INST environment variable (and other FishEye-specific environment variables) in the <FishEye install directory>\wrapper\conf\wrapper.conf file, following the [instructions](#) below.
5. Install FishEye as a service as follows:
 - a. Open an Administrator command prompt by searching for 'Command prompt' in the Windows Start menu, right-clicking on **Command Prompt** and then choosing **Run as administrator**.
 - b. Change directory to <FishEye install directory>\wrapper\bin and run Fisheye-Install-NTService.bat. If you run into any problems starting the wrapper, you'll find its logs in <FishEye install directory>\var\log\wrapper.log.
6. Start the Fisheye service under the Windows Control Panel; you can search in the Start menu for 'services', and in the list of services, right-click on the 'FishEye' item and choose **Start**. You can also stop the FishEye service in this way.

i Please note that:

- If you make changes to the wrapper.conf file, having already started the service, you need to stop and then restart the service for it to make use of the changed configuration.
- If in future you move the FishEye install directory, you will need to uninstall (using Fisheye-Uninstall-NTService.bat) and then reinstall the FishEye service.

Setting FishEye environment variables for Windows Services

Please note, that if you run FishEye as a Windows service, any FishEye-specific [environment variables](#) must be set in your <FishEye install directory>\wrapper\conf\wrapper.conf file.

If you run into any problems starting the wrapper, you'll find its logs in <FishEye install directory>\var\log\wrapper.log.

If there are other Java parameters you wish to add, then you will need to add them under the additional parameters section, e.g.

```
# JDK Additional Parameters for jmx
wrapper.java.additional.4=-Dcom.sun.management.jmxremote
wrapper.java.additional.5=-Dcom.sun.management.jmxremote.port=4242
wrapper.java.additional.6=-Dcom.sun.management.jmxremote.authenticate=false
wrapper.java.additional.7=-Dcom.sun.management.jmxremote.ssl=false
wrapper.java.additional.8=-Dcom.sun.management.jmxremote.password.file=.
/wrapper/jmxremote.password
wrapper.java.additional.9=-Dwrapper.mbean.name="wrapper:type=Java
Service Wrapper Control"

wrapper.java.additional.10=-XX:PermSize=256m
```

To add the FISHEYE_INST environment variable, the Java MaxPermSize parameter, or the -Xrs options, use the following:


```
wrapper.java.additional.11=-Dfisheye.inst="c:/path/to/FISHEYE_INST"  
wrapper.java.additional.12=-XX:MaxPermSize=128m  
wrapper.java.additional.13=-Xrs
```

Note that the the -Xrs options should be used when running FishEye as a service under Windows to prevent the JVM closing when an interactive user logs out.

Your memory settings can also be found in this file:

```
# Initial Java Heap Size (in MB)  
wrapper.java.initmemory=256  
  
# Maximum Java Heap Size (in MB)  
wrapper.java.maxmemory=1024
```

Increase these values if you have a large repository or expect to use more memory (init of 256, and a max of 1024 are the default values).

In Fisheye/Crucible 1.6.4 and higher, you can check the JVM input arguments by clicking **System info**, under 'System Settings' in the admin area.

Troubleshooting

Extracting files from wrapper.zip

Some customers have reported trouble running the wrapper. These can be avoided by:

- Uncompressing wrapper.zip with Winzip or WinRar rather than using the **Extract All** command in the Windows right-click contextual menu.
- If the wrapper.zip filename appears green instead of black in Windows Explorer, decrypt it, prior to unzipping its contents, by right-clicking on the file, choose **Properties**, click the **Advanced** button, then clear the **Encrypt contents to secure data** checkbox.

Warning when using 64-bit Java JDK

When using a 64-bit Java JDK with the wrapper obtained via the link in the install instructions above, you will see the following in the wrapper.log file, and the service won't start:

```
WARNING - Unable to load the Wrapper's native library 'wrapper.dll'. The file is located on the path  
at the following location but could not be loaded:  
C:\install\service\fisheye28\wrapper\lib\wrapper.dll.
```

```
Please verify that the file is readable by the current user and that the file has not been corrupted in  
any way. System signals will not be handled correctly.
```

This is caused by using a 64-bit JDK (even on a 64-bit machine). Changing to a 32-bit version of the JDK will prevent this warning. Community Edition versions of the 64-bit Windows Java Service Wrapper are not currently available.

Wrapper configuration and "-server" parameter

Please note that the wrapper configuration provided above uses the -server parameter to enable the [Java HotSpot\(TM\) Server VM](#). This feature is only available if you use the JDK. If you use the JRE you will likely get the following error in your logs:

```
INFO | jvm 1 | 2010/12/20 18:19:28 | Error: missing `server' JVM at `C:\Program  
Files\Java\jre6\bin\server\jvm.dll'.
```

A common issue is that customers remove the -server parameter from the wrapper.conf file. Please note

that if you do this, the wrapper script will ignore any of the following JVM parameters in the file unless you change the sequence to be in order, starting from wrapper . java . additional . 1. This is an issue with the wrapper application.

In this situation it's best to install and run FishEye/Crucible with the JDK to get all the advantages of the - server functionality. You also need to force the wrapper to use the JDK by specifying the path to the Java JDK in the wrapper . conf file, as described in the installation instructions above.

Configuring indexing

Understanding how FishEye indexes repositories

FishEye's administration interface allows you to configure the number of threads used for repository indexing.

FishEye uses two types of indexing operation:

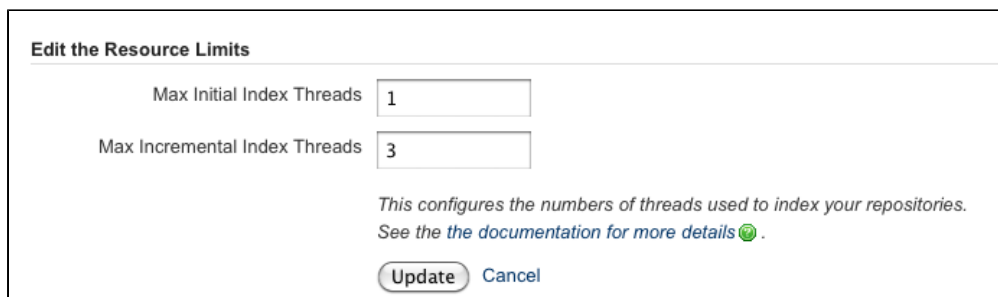
- **Initial indexing** - the complete scan of the repository that is performed when FishEye first accesses the repository, and
- **Incremental indexing** - the quick, background process that regularly scans repositories for changes.

Configuring the number of indexing threads

To configure the number of threads FishEye uses for indexing:

1. Go to the FishEye admin area and click **Server** (under 'Global Settings').
2. Click **Edit Details** (under 'Resource Limits'). You can change the number of threads available for each type of indexing operation.
3. Click **Update**.

Screenshot: FishEye multi-threading configuration



Edit the Resource Limits

Max Initial Index Threads

Max Incremental Index Threads

This configures the numbers of threads used to index your repositories.
See the the documentation for more details [🔗](#).

Choosing a thread setting

Changing the number of threads always involves a trade-off in performance; you should generally change these settings in increments of one, then observe the performance impact. A safe range is from 1-3 threads on both settings. Using more than three threads on either setting is not recommended; the more threads, the busier the server will become when indexing. There is certainly no benefit in having more threads than you have repositories. Adding a lot of threads may impact overall system performance.

By default, FishEye uses one thread each for the two indexing types.

Note that decreasing the number of indexing threads does not have an effect until any current scanning ends. To force this effect immediately, perhaps because of a performance issue, a FishEye restart is needed.

For more information on managing FishEye's performance, see the [FishEye Tuning](#) page.

Optimizing the Search Indexes on Restart

FishEye/Crucible allows you to optimize the search indexes when you restart your instance.

You can use this feature to:

- Reduce the size of the search indexes,
- Make the search faster if the indexes are significantly under-optimized, and

- Reduce the number of "Too many open files" [errors](#) that you might see in the logs. This is useful on systems where FishEye or Crucible have constraints on the number of files they can open.

In most circumstances, the search indexes should be fairly close to their optimal state. This is usually the case when using Crucible with FishEye, or when using FishEye alone. The search technology we use, [Lucene](#), results in indexes fairly close to optimal when most of the operations on the indexes are [additions](#) (the additions are the changesets that are indexed as they appear in your repositories). For this reason, it should not be necessary to run the optimization often.

When you are using this feature, the optimization will run solely on the *next start*. It will not run every time FishEye or Crucible starts, to prevent slowing the start of your instance. The optimization itself should be fast, and within 10 minutes even for large indexes.

Running the Optimization

1. Navigate to the FishEye/Crucible administration console.
2. Click the **'Shutdown'** link in the left menu. Note, you will need to confirm the shutdown at a confirmation screen before your instance is shut down.
3. In the window that appears, click the **'Optimize the search indexes the next time that FishEye and Crucible start'** checkbox (or **'Optimize the search indexes the next time that FishEye/Crucible starts'** if you are using either FishEye or Crucible alone).
4. Click **'Shutdown'**. Your instance will be shut down, and the optimization of the search indexes will be scheduled for the next startup.

Customising FishEye's look and feel

Note that the [Atlassian Support Offering](#) does not cover the customizations described on this page. Assistance may be obtained through the Atlassian community from answers.atlassian.com or from an [Atlassian Expert](#).

FishEye commercial/academic license users have access to the HTML/JSP source of FishEye and can customize FishEye's look and feel. If you have a commercial license assigned to your account, you will see a **source download** option on your download page.

FishEye Source Edition

To use custom HTML/JSP content, you must be using a build of FishEye that contains the JSP source. These builds are named `fisheye-1.x.y-jspsource.zip` instead of the normal `fisheye-1.x.y.zip` bundle.

You must install a JSP source build – you cannot copy JSP files from a source build into an existing ordinary installation.

Customizing Content

With the JSP source build you can modify any of the files in `<FishEye installation directory>/content/`. However we strongly recommend that you use separate installation and `FISHEYE_INST` directories (as described in [Installing FishEye on Windows](#) and [Installing FishEye on Linux and Mac](#)), and that you store your modified files in `FISHEYE_INST/content` (If you use `FISHEYE_INST/content`, you only need to keep your modified HTML/JSP files in that directory. This simplifies your upgrade process).

When you make changes to content, your changes should appear when you next refresh the page in your browser. If they do not, then log in to the FishEye Administration screens, click **Content** under 'System Settings', and follow the instructions on the 'Content' page.

Content



Note: these functions are intended for people with the JSP-Source Edition.



When updating JSPs or other content, you may need to **flush** before that content is served by FishEye and Crucible. In some situations, such as deleting an existing file from your custom content, a **flush** may not be sufficient. In such cases:

- Stop FishEye and Crucible.
- Delete the JSP classes directory: `/opt/j2ee/domains/atlassian.com/eac-fisheye/data/var/tmp/web`
- Restart FishEye and Crucible.

Custom content directory `/opt/j2ee/domains/atlassian.com/eac-fisheye/data/content`

Flush

An alternative way of customizing FishEye's and Crucible's look and feel is to edit specific CSS files. Please refer to [this Knowledge Base article](#) for some examples.

Environment variables

Environment variables are system-wide settings that are required for certain applications.

On this page:

- [Setting environment variables under Windows XP](#)
- [Setting environment variables under Linux or Mac](#)

Please note:

- FishEye 3.4 and later versions: If you used the installer to set up FishEye as a Windows service, you can change JVM settings using the tool provided with the installer. See [Setting JVM system properties](#).
- Java Service Wrapper: If you set up FishEye as a Windows service using the Java Service Wrapper, any environment variables that you require must be set in your `<FishEye install directory>/wrapper/conf/wrapper.conf` file. See [Running FishEye as a Windows service](#).

FishEye uses the following environment variables:

Environment variable	Notes
JAVA_HOME	FishEye uses this to select the Java Virtual Machine (JVM) to use. If this environment variable is not set, FishEye will use whatever Java executable is available on the path. In Linux systems, this may sometimes be GCJ-based which causes some problems running FishEye. We recommend that JAVA_HOME should point to the root folder of the Java installation (e.g. <code>/usr/jdk/jdk1.8.0_60</code>). See either Installing FishEye on Windows or Installing FishEye on Linux and Mac for further instructions on setting JAVA_HOME. If you're using the Java Service Wrapper to run FishEye as a service on Windows, the wrapper will use the Java version found in PATH instead of using JAVA_HOME.

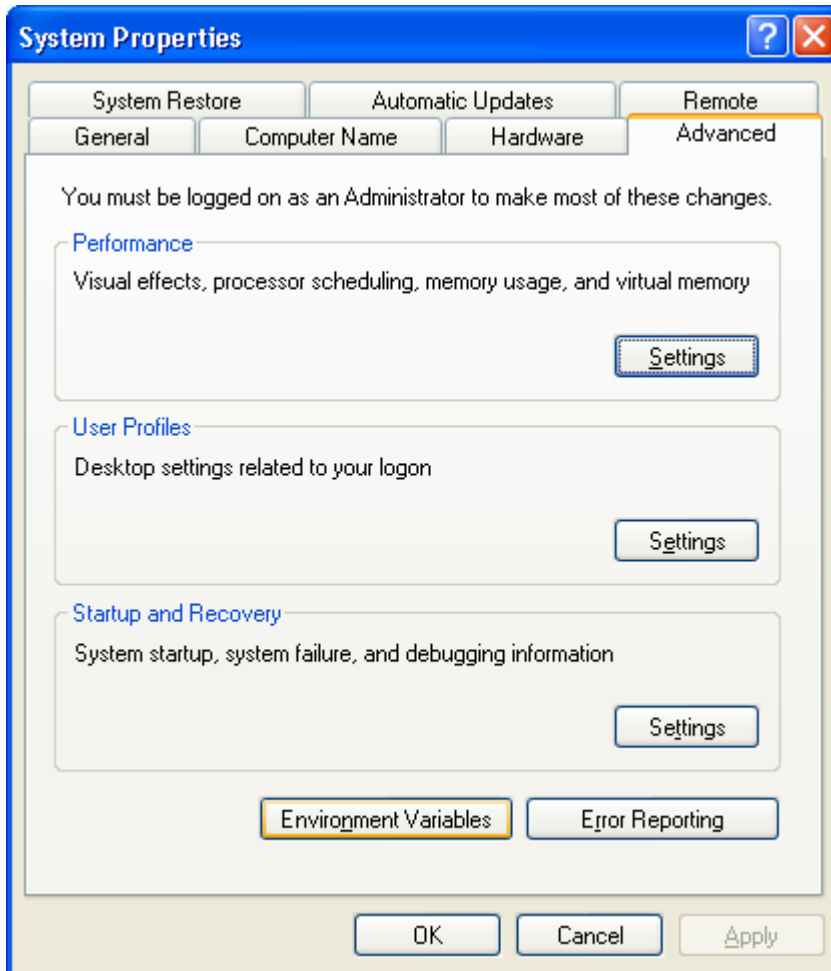
FISHEYE_OPTS	Use this to pass parameters to the Java Virtual Machine (JVM) that runs FishEye. This is typically used to set the Java heap size available to FishEye. See: • How FishEye uses memory • Fix out of Memory Errors For example, with an Oracle JVM, you could increase the max heap by using: <pre>FISHEYE_OPTS=-Xmx2048m -XX:MaxPermSize=128m</pre> This would give FishEye a max of 2048 MB heap and a Max permanent generation size of 128MB. See Tuning FishEye performance for more information. You can also use FISHEYE_OPTS to set various system properties in FishEye. For example: <pre>FISHEYE_OPTS=-Dname.of.property=value</pre> After having set FISHEYE_OPTS and restarting your server, ensure that your changes have been registered by navigating to Administration > Sys Info/Support > System Info and checking the output under "JVM Input Arguments".
FISHEYE_ARGS	See Command-line options. FISHEYE_ARGS are the arguments which will be passed to FishEye when it is started. You can set this to --debug, for example, or --debug-perf if you always want to have FishEye debugging put into the FishEye log files.
FISHEYE_LIBRARY_PATH	Used to tell FishEye where it should look to load any additional native libraries.
FISHEYE_INST	Used to tell FishEye where to store its data. It is recommended that you separate FishEye's data from its application files in <FishEye install directory> by using this variable. For example: <pre>c:\path\to\fisheye_data (no trailing backslash)</pre>  Read more about using FISHEYE_INST in either Installing FishEye on Windows or Installing FishEye on Linux and Mac.

Setting environment variables under Windows XP

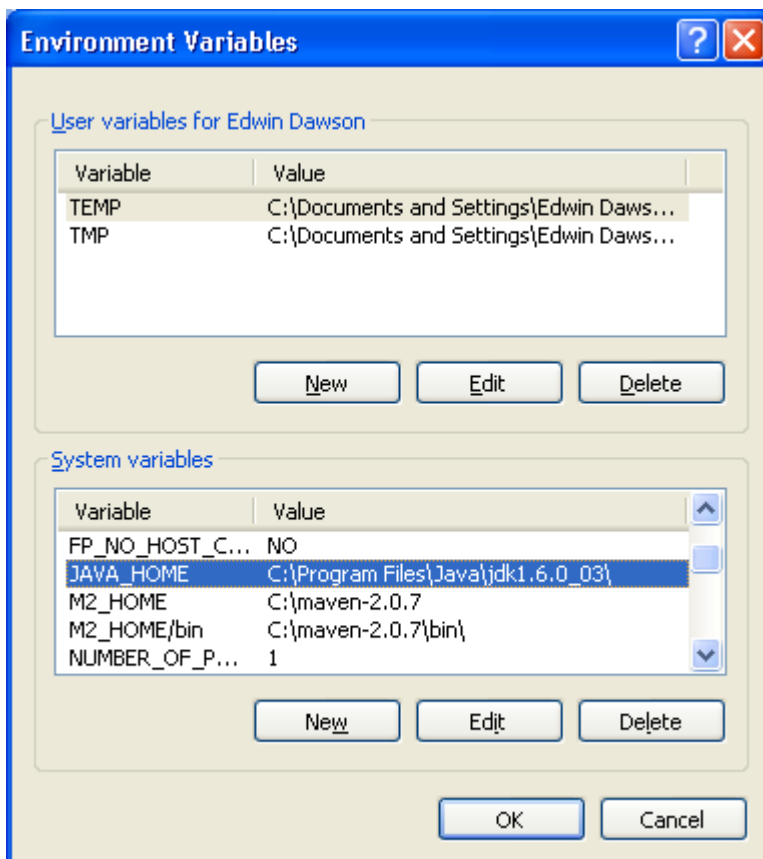
(System environment variables are ignored if FishEye is run as a Windows service. Please refer to [Running FishEye as a Windows service](#).)

(Linux instructions for setting environment variables are [here](#).)

1. Click **Start > Control Panel > System**.



2. Click the **Advanced** tab.
3. Click the **Environment Variables** button.



4. Click **New**.
5. In the **Variable name** field, enter the name of the environment variable, for example:

```
FISHEYE_OPTS
```

6. In the **Variable value** field, enter the setting as required. This may be quite cryptic, for example the default value for FISHEYE_OPTS is this:

```
-Xmx256m
```



7. Restart the computer.

Setting environment variables under Linux or Mac

There are a number of ways to set environment variables on Linux or UNIX based systems (including **Mac OS X**). Here are just two:

For your current user:

1. Open up a shell or terminal window
2. Type this command:

```
vi ~/.profile
```

(vi is a text editor, you can use another if desired)

3. Add this command:

```
export (variable name)=(variable value)
```

Where (variable name) and (variable value) are the environment variable elements. For example, if the environment variable you are setting is FISHEYE_OPTS, and the variable value is -Xmx256m, you would type the following:

```
export FISHEYE_OPTS=-Xmx256m
```

Add this command on its own line at the end of the file.

4. Save, and restart your shell.

For all users in the system:

1. Open up a shell or terminal window
2. vi /etc/profile (replace vi with your favorite text editor)
3. Add export (variable name)=(variable value) on its own line at the end of the file
4. Save, and restart your shell

If you are using a GUI, you may not need to open up the shell. Instead, you might be able to open the file directly in a graphical text editor.

i If you are experiencing memory errors in FishEye, see [Fix Out of Memory errors](#).

Configuring outgoing web proxy support for FishEye or Crucible

Accessing external hosts

FishEye may need to connect to hosts that are outside your network. For example, in Administration > Server Settings > Update Notifications, FishEye needs to access <http://updates.atlassian.com> to check for updates. If the FishEye server can't access the site it will throw the following error:

```
ERROR - Error while checking for newer versions from
http://update.atlassian.com (update.atlassian.com)
```

When trying to check if there are newer versions of FishEye or Crucible

In some environments, access to hosts outside the network are forbidden (in which case, you will need to manually check for updates). In other environments access to hosts outside of the network need to go through a proxy.

If you do have a proxy server, you need to tell FishEye how to send its requests through the proxy.

Configuring an outbound HTTP proxy in FishEye

Proxy support is configured by passing certain arguments to the Java Virtual Machine on startup (for FishEye, you can add them to the FISHEYE_OPTS [Environment Variables](#)). These properties follow the [conventions defined by Sun](#):

- `http.proxyHost`
- `http.proxyPort` (default: 80)
- `http.nonProxyHosts` (default: <none>)
- `https.proxyHost`
- `https.proxyPort`

At a minimum, you need to define `http.proxyHost` to configure an HTTP proxy, and `https.proxyHost` for an HTTPS proxy. System property configuration is described on the [JVM system properties](#) page.

Properties `http.proxyHost` and `http.proxyPort` indicate the proxy server and port that the HTTP protocol handler will use.

```
-Dhttp.proxyHost=proxy.example.org -Dhttp.proxyPort=8080
-Dhttps.proxyHost=proxy.example.org -Dhttps.proxyPort=8080
```

Property `http.nonProxyHosts` indicates the hosts which should be connected to directly and not through the proxy server. The value can be a list of hosts, each separated by a `|`, and in addition a wildcard character (`*`) can be used for matching. For example:

```
-Dhttp.nonProxyHosts=*.foo.com|localhost
```

Note: You may need to escape the pipe character (`|`) in some command-line environments.

If the `http.nonProxyHosts` property is not configured, all web requests will be sent to the proxy.

Configuring HTTP proxy authentication

Proxy authentication is also configured by providing [system properties](#) to Java in your application server's configuration file. Specifically, the following properties:

- http.proxyUser – username
- http.proxyPassword – secret
- https.proxyUser
- https.proxyPassword

How it looks

So, an example of your FISHEYE_OPTS [Environment Variables](#) will be:

```
FISHEYE_OPTS="-Xms128m -Xmx1024m -XX:MaxPermSize=128m
-Dhttp.proxyHost=proxy.example.org -Dhttp.proxyPort=8080
-Dhttp.nonProxyHosts=*.foo.com|localhost -Dhttp.proxyUser=USERNAME
-Dhttp.proxyPassword=SECRET -Dhttps.proxyHost=proxy.example.org
-Dhttps.proxyPort=8080 -Dhttps.proxyUser=USERNAME
-Dhttps.proxyPassword=SECRET"
```

After having set the FISHEYE_OPTS and restarting your server, go to Administration > Sys Info/Support > System Info, and check your JVM Input Arguments to ensure that your server is picking up your FISHEYE_OPTS as expected.

Using AJP with FishEye

When using AJP authentication, only the authentication is delegated to the proxying web server.

The configured [user directories](#) are still used to hold the user, groups and memberships lists. In most configurations you should use an internal directory together with AJP authentication.

If possible you should use a supported [external directory](#), rather than AJP authentication.

You can use the AJP protocol to proxy inbound requests from a web server through to an application server that sits behind the web server.

AJP expects requests to be pre-authenticated via an external server before arriving at FishEye. Typically, this would be a web server (e.g. Apache) configured to perform password and role checking for a given URL. If successful, the server forwards the request to the FishEye server using the AJP protocol.

FishEye configuration

For FishEye to use AJP authentication, the following two values must be configured:

- The AJP Bind Address must be set per FishEye instance. See also [Server Settings](#).
- The user's Auth Type must be set to 'ajp'.

Apache configuration

mod_proxy_ajp

To configure Apache HTTP Server to forward all requests for /fisheye to a FishEye instance with an AJP Bind address of `localhost:8009`, on the same machine, ensure that the Apache [proxy_ajp_module](#) module is enabled, and add these lines to the VirtualHost you wish to serve FishEye through:

```
ProxyPass /fisheye/ ajp://localhost:8009/fisheye/ nocanon
```

mod_jk

To configure Apache HTTP Server so that all requests to Apache for the path /fisheye are forwarded to a FishEye instance on the same machine with an AJP bind address of `localhost:8009`, add these lines to

your Apache configuration:

```
LoadModule jk_module modules/mod_jk.so

JkWorkersFile /path/to/workers.properties
JkLogFile /var/log/mod_jk.log
JkLogLevel debug
JkLogStampFormat "[%a %b %d %H:%M:%S %Y] "
JkMount /fisheye/* worker1

JkOptions +ForwardURICompatUnparsed
```

Then, create a file under `/path/to/workers.properties` and add:

```
worker.list=worker1
worker.worker1.type=ajp13
worker.worker1.host=localhost
worker.worker1.port=8009
```

Custom authentication

When using custom authentication, only the authentication (and the per-repository restrictions) are delegated to the configured class.

The configured [user directories](#) are still used to hold the user, groups and memberships lists. In most configurations you should use an internal directory together with custom authentication.

If possible you should use a supported [external directory](#), rather than custom authentication.

To implement an arbitrary form of authentication and authorization for FishEye you need to provide a class which extends `com.cenqua.fisheye.user.plugin.AbstractFishEyeAuthenticator`. You can find more information about custom FishEye authorization in the [online javadocs](#) and the [library jar](#).

For FishEye to use the authenticator, it must be compiled, placed in a jar archive and then put in the `$FISHEYE_INST/lib` directory. If other third-party libraries are required by your authenticator, they must also be in the `$FISHEYE_INST/lib` directory.

Global configuration

After implementing a custom authenticator, the next step is to configure FishEye to use it.

Click **Setup Custom authentication** on the FishEye **Authentication** page, in the Admin area.

You will be presented with a form containing the following fields to be set:

Classname	The fully qualified class name of your <code>AbstractFishEyeAuthenticator</code> , e.g. <code>com.cenqua.fisheye.user.plugin.ExampleFishEyeAuthenticator</code> .
Cache TTL (positive)	How long FishEye should cache permission checks. Example values are: 0 secs, 5 mins.
Auto-add	FishEye can automatically create a user it has not previously encountered if the user can successfully authenticate against your authenticator.

Properties	Any properties your authenticator requires. These will be passed to its <code>init()</code> method. This field should comply with the <code>java.util.Properties</code> format. Example: <pre># comments name1=value1 name2=value2</pre>

Per-repository constraint configuration

You may also require a per-repository constraint to restrict access to specific repositories using your custom authenticator. If a custom authenticator is set, then the [Permissions Summary table](#) will display the constraint per repository and a link to enable you to edit it.

When using FishEye's built-in groups in conjunction with a custom authenticator, a user will have access to a repository if:

1. the user is in an allowed group for the repository
2. or the `hasPermissionToAccess` method of the custom authenticator returns true.

The 'Authentication Test' page allows you to enter a user's credentials and to test the user's authentication. It will also test which repositories the user is authorized to access.

Configuring SSL cipher suites for Jetty

You may wish to set the cipher suites and protocols that are used for a specific SSL connector when Jetty starts up:

- To *include* a cipher suite or protocol you require that is not enabled by default.
- To *exclude* a cipher suite or protocol that is considered too weak to use, or for which a vulnerability has been discovered.

The Java Virtual Machine provides the SSL cipher suites that Jetty uses. See the [JSSE Provider documentation](#) for more information about the available cipher suites.

Note that for FishEye 3.6, and later, cipher suites and protocols are now defined in the `config.xml` file. For FishEye 3.5, and earlier versions, cipher suites were defined in the `jetty-web.xml` file – see [Configuring SSL cipher suites for Jetty](#).

Enabling cipher suites or protocols

You can specify the cipher suites or protocols that the Jetty webserver (bundled with FishEye) will use:

1. Shut down FishEye.
2. Open the `config.xml` file in your FishEye instance directory (the data directory that the `FISHEYE_INST` system environment variable points to).
3. Find the `<ssl>` element under the `<web-server>` element in the file, and add `<includeCipherSuites>` and `<includeProtocols>` as needed. For example:

config.xml

```
<config version="1.0">
  <web-server context="/foo">
    <ssl bind=":443" keystore="/etc/dev/keystore"
keystore-password="" truststore="/etc/dev/keystore"
truststore-password="">
      <includeProtocols>
        <protocol>TLSv1</protocol>
        <protocol>TLSv1.1</protocol>
        <protocol>TLSv1.2</protocol>
      </includeProtocols>
      <includeCipherSuites>

        <cipherSuite>TLS_DHE_DSS_WITH_AES_128_CBC_SHA</cipherSuite>

        <cipherSuite>TLS_DHE_RSA_WITH_AES_256_CBC_SHA</cipherSuite>

        <cipherSuite>TLS_RSA_WITH_AES_256_CBC_SHA</cipherSuite>
      </includeCipherSuites>
    </ssl>
  </web-server>
```

4. Restart FishEye.

This will cause the Jetty SSL connector to only use the cipher suites and protocols specified in the xml.

Disabling cipher suites or protocols

You can exclude a cipher suite or protocol from those that the Jetty webserver (bundled with FishEye) will use. You may want to do this for a suite or protocol that is considered too weak to use, or for which a vulnerability has been discovered. Note that Jetty performs the exclude operation after the include operation. Therefore, if a cipher suite is both included and then excluded as part of the same configuration, it is disabled.

1. Shut down FishEye.
2. Open the `config.xml` file in your FishEye instance directory (the data directory that the `FISHEYE_INST` system environment variable points to).
3. Find the `<ssl>` element under the `<web-server>` element in the file, and add `<excludeCipherSuites>` and `<excludeProtocols>` as needed. For example:

config.xml

```

<config version="1.0">
  <web-server context="/foo">
    <ssl bind=":443" keystore="/etc/dev/keystore"
    keystore-password="" truststore="/etc/dev/keystore"
    truststore-password="">
      <excludeProtocols>
        <protocol>SSLv3</protocol>
      </excludeProtocols>
      <excludeCipherSuites>

<cipherSuite>SSL_RSA_WITH_3DES_EDE_CBC_SHA</cipherSuite>

<cipherSuite>SSL_DHE_RSA_WITH_DES_CBC_SHA</cipherSuite>
  <cipherSuite>SSL_DHE_DSS_WITH_DES_CBC_SHA</cipherSuite>
    </excludeCipherSuites>
  </ssl>
</web-server>

```

4. Restart FishEye.

This will cause the Jetty SSL connector to use all the cipher suites and protocols provided by the JVM, except the ones specified in the xml.

JVM system properties

Administrators can configure the Java Virtual Machine (JVM) system properties described on this page to control low level aspects of FishEye and Crucible's operation. The application must be restarted on all platforms for the changes to become effective. After the application starts up you can verify the setting by using the command-line tool `jinfo <pid>` or look up "JVM input arguments" on the Administration > System Info page.

system.properties (all platforms)

The recommended approach is to add the property to the `FISHEYE_INST/system.properties` file. Add this as a new text file, using the Java [.properties format](#), if it doesn't already exist.

Environment Variable (all platforms)

Properties can also be set by passing them with the `-D` parameter to the JVM using the `FISHEYE_OPTS` [environment variable](#).

Windows Service

If you used the FishEye or Crucible installer for Windows, you can edit JVM properties as follows.

FishEye/Crucible 3.4.4+

Edit JVM properties using the tool included with the installer. Launch the tool by going to one of:

- **Start > All Programs > FishEye > Configure FishEye**
- **Start > All Programs > Crucible > Configure Crucible**

Ensure that you restart the FishEye or Crucible service after changing the

See also:

- [Command-line options](#)
- [FishEye installation guide](#)
- [Setting JAVA_HOME](#)
- [Best practices for FishEye configuration](#)

JVM properties.

Earlier versions:

Edit JVM properties using this tool:

<http://www.apache.org/dist/commons/daemon/binaries/windows/commons-daemon-1.0.15-bin-windows.zip>

To use it, you need to rename `prunmgr.exe` from the zip file to `Atlassian FishEye.exe` or `Atlassian Crucible.exe` depending on which installer you used. Then JVM properties such as `Xmx` and `XX:MaxPermSize` can be edited on the **Java** tab.

Do not reference any environment variables in the settings (e.g. `%FISHEYE_INST%`). Instead, use the absolute path.

- System properties that only relate to Crucible have names beginning with `crucible`.
- When integrating FishEye/Crucible with Single Sign On applications, you may require an adjustment to the JVM properties, specifically the HTTP header size. This may also be useful in other circumstances where FishEye's default HTTP header is too small (at 4096 bytes).

On this page:

- [General properties](#)
- [FishEye pipeline indexing](#)
- [Git manifest](#)
- [Mail client](#)

General properties

Property	Legal values	Default values	Description
<code>fisheye.blame.calc.enable</code>	true/false	true	Controls whether blame calculations are calculated during indexing. Note that blame calculation during indexing is disabled if Store Diff Information is disabled. See store diff info . See Blame calculations for details.

<code>fisheye.blame.scm.fallback.enable</code>	true/false	false	Controls whether blame data is retrieved from the SCM during indexing (when blame couldn't be based on ancestors due to indexing). See Blame calculations for details.
<code>fisheye.enable.request.blame.calculation</code>	true/false	3.10: false 3.9: true	Controls whether user can request blame data from the SCM when viewing the file or diff (when blame couldn't be calculated and retrieved during indexing). See Blame calculations for details.
<code>fisheye.ldap.sync.page.size</code>		1000	Controls the page size used when FishEye requests data from LDAP user directories. The LDAP server must support LDAPv3. A value of 0 disables pagination.
<code>crucible.detect.metadata.revision.changes</code>	true/false	3.0: false 2.6: true	If true, detect and disables creation of reviews from changesets which contain metadata only changes (e.g. SVN property changes). See here for more details.

<code>crucible.review.content.size.limit</code>	number of file revisions allowed in a review	800	Limits the "size" of a review to prevent extremely large reviews accidentally being created and potential slowing the instance down.
<code>fisheye.branch.recently.active.check.limit</code>	number of branches	250	Used in CommitGraph control sorting branches. If more than the limit, sorted alphabetically otherwise sorted by recent activity.
<code>fisheye.changeset.paths.limit</code>	number of paths	10000	This setting is to prevent extremely large changesets (which are often the result of a merge error) from slowing FishEye/Crucible down. See Indexed paths truncated at 10000 for the messages that are logged if the value is reached.
<code>fisheye.git.http-backend-timeout</code>	Integer milliseconds	3600000	This property sets the amount of time that the FishEye service should allow the underlying Git process to execute when attempting to serve a Git push or pull request.

<code>fisheye.p4.disable.ancestry</code>	true/false	false	If true, stops FishEye from processing P ancestry information. Change lists still be assign to branches. you do not use stable Perform branchespecs manage branches, set this to true.
<code>fisheye.p4.disable.branchspecs</code>	true/false	false	If true, stops FishEye from reading Perform branch specs. All change lists will appear on the 'head' branch. If you not use stable Perform branchespecs manage branches, set this to true.
<code>fisheye.search.max.repositories</code>	number of repositories	100	Used by FishEye to limit the search to top <i>n</i> repositories (by alphabetical order) when a cross-repository search is performed.
<code>jetty.http.headerbuffersize</code>	bytes	4096	This controls size of the largest HTTP header value that FishEye allow (through its embedded servlet engine Jetty). Some authentication systems, such as Single Sign On application require larger header value

<code>org.eclipse.jetty.server.Request.maxFormContentSize</code>	bytes	3000000	Max form content size - increase this value if your push fails with HTTP error 4 (see this KB article: Git PUSH results: RPC failed, result=22, HTTP code = 413)
<code>crucible.users.per.group.in.review</code>		10	Maximum number of users that will be added from a group when adding a group to a review
<code>fisheye.lucene.boolean.query.max.clause.count</code>	integer	10000	Maximum number of clauses in a Lucene boolean query. This limit is used while rendering large lists (e.g. projects, user reviews dashboard). You may need to increase the limit if you have a problem as in RUC-7794 .

FishEye pipeline indexing

See [Pipelined indexing](#).

Property	Legal values	Default Value	Description	FishEye version
<code>fisheye.pipeline.batch.cslimit</code>	1 - 50000	10000	The pipeline processes changesets in batches. A bigger batch size reduces the number of calls to the underlying SCM, but also increases heap consumption and reduces the concurrency of the pipeline. Smaller batch sizes increase the concurrency of the pipeline, reduce heap consumption, but result in more calls to the underlying SCM.	3.0.0+

<code>fisheye.pipeline.batch.pathlimit</code>	1000 - 120000	60000	The pipeline processes changesets in batches. This property sets the maximum allowed number of unique paths in a batch (summed across changesets). A higher value will reduce the number of calls to the underlying SCM, at the expense of more heap consumption.	3.0.0+
<code>fisheye.pipeline.fairness</code>	true, false	true	If true, repositories are processed in a round-robin fashion by the pipeline. This means that all repositories currently indexing will incrementally progress. If false, the pipeline always favors newer changesets, regardless of which repository they belong to. This means that repositories with newer changesets will take priority over repositories with older changesets. This results in faster processing for newer repositories, but with the possibility of stalling indexing of repositories with older commits.	3.0.0+
<code>fisheye.pipeline.threads</code>	4-1000	The number of processor cores detected, or 4, whichever is greater	The number of threads available to work on indexing tasks. A higher value allows for potentially greater concurrency in indexing activities, but will increase the load on the server and the load on your SCM.	3.0.0+

Git manifest

Property	Legal values	Default value	Description	Version
<code>fisheye.manifest.forceupgrade</code>	true or false	true	Controls whether manifests are upgraded before repository indexing continues or as a background process as time permits during regular indexing	3.4.0+

<code>fisheye.manifest.maxdepth</code>	1-100	3	Controls how many delta manifests are written in a chain before a full manifest is written	3.4.0+
<code>fisheye.manifest.maxupgradepaths</code>	100 upwards	500000	The maximum number of paths that will be upgraded before a batch is committed to the FishEye index	3.4.0+
<code>fisheye.manifest.upgradebatch</code>	10 upwards	1000	When performing background upgrading this is the minimum number of changesets that will be upgraded in each indexing poll period.	3.4.0+

Mail client

You can tune the mail client in FishEye with system properties.

The available system properties are listed and described at <https://javamail.java.net/nonav/docs/api/com/sun/mail/smtp/package-summary.html>.

For example, `mail.smtp.quitwait=false` can be used in cases where the FishEye mail server doesn't respond to a QUIT message, and causes the mail client to hang.

Configuring user managed mappings

In FishEye and Crucible, Administrators can control whether users can use the [Author Mapping](#) setting to map their own FishEye/Crucible usernames to repository committer accounts or not. By default, the setting allows users to set their own mappings.

If you wish to lock down the mappings for security or audit reasons, this setting lets you restrict all management of mappings to FishEye/Crucible administrators only.

To do this, click **Administration** in the footer of the FishEye/Crucible interface and then **Authentication** (under 'Security Settings') in the left navigation bar. You can set **User Managed Mappings** either **On** or **Off**. The setting is applied immediately.

Built-in

Public Signup ?	OFF (Turn On)
Use CAPTCHA for Signup ?	ON (Turn Off)
User managed mappings	ON (Turn Off)

Configuring FishEye security

Choose from the pages below for specific information on FishEye security settings:

- [Best practices for configuring FishEye security](#)
- [Brute force login protection](#)
- [Configuring CAPTCHA](#)
- [Granting administrator privileges](#)
- [Configuring public signup](#)

Best practices for configuring FishEye security

The best way to harden a system is to look at each of the involved systems individually. Contact your company's security officer or department to find out what security policies you should be using. There are many things to consider, such as the configuration of your underlying operating systems, application servers, database servers, network, firewall, routers, etc. It would be impractical to outline all of them here.

This page contains guidelines on good security practices, to the best of our knowledge.

Configuring the web server

Please refer to the following guides for system administrators:

- How to configure Apache to lock down the administration interface to those people who really need it. See [Using Apache to limit access to the Confluence administration interface](#) for guidance.
- How to reduce the risk of brute force attacks: [Configuring CAPTCHA](#).

Configuring the application server

See the following system administrator guide for general hints at the application server level (FishEye uses Jetty, but the principles described here still apply):

- [Tomcat security best practices](#)

Configuring the application

The way you set up FishEye roles, permissions and processes makes a big difference in the security of your FishEye site.

Below are some more FishEye-specific items to consider. None of these provides 100% security. They are measures to reduce impact and to slow down an intruder in case your system does become compromised.

- Restrict the number of users with powerful roles or group memberships. If only one department should have access to particularly sensitive data, then do restrict access to the data to those users. Do not let convenience over-rule security. Do not give all staff access to sensitive data when there is no need.
- Put documented procedures in place for the case of employees leaving the company.
- Perform security audits regularly. Know who can help in case a security breach occurs. Perform 'what if' planning exercises. ('What is the worst thing that could happen if a privileged user's password were stolen while he's on vacation? What can we do to minimize damage?').
- Make sure the FishEye database user (and all datasource database users) only has the amount of database privileges it really needs.
- Monitor your binaries. If an attacker compromises an account on your system, he will usually try to gain access to more accounts. This is sometimes done by adding malicious code, such as by modifying files on the system. Run routine scripts that regularly verify that no malicious change has been made.

Securing the FishEye instance directory

For production use, we strongly recommend that you secure the FishEye instance directory against unauthorized access.

The location of the FishEye instance directory is defined by the FISHEYE_INST system variable; we highly recommend that this be a completely separate location from the <FishEye home directory>(the install location). See [FishEye folder layout](#) for details of the data kept in the instance directory.

We recommend the following precautions:

- Assign a separate restricted user account on the machine for running FishEye (not a root/administrator user).
- Ensure that only the user running FishEye can access the FishEye instance directory, and that this user has read, write and execute permissions, by setting file system permissions appropriately for your operating system.

Configuring system admin access

Below are some things to consider specifically related to the system admin role:

- Keep the number of FishEye administrators extremely low. For example, 3 system administrator accounts should be the maximum.
- The administrators should have separate FishEye accounts for their administrative roles and for their day to day roles. If John Doe is an administrator, he should have a regular user account without administrator access to do his day to day work. This could be a 'john.doe' account. In addition, he should have an entirely separate account (that cannot be guessed by an outsider and that does not even use his proper name) for administrative work. This account could be 'jane smith' – using a username that is so obscure or fake that no outsider could guess it. This way, even if an attacker singles out the actual person John Doe and gets hold of his password, the stolen account would most likely be John's regular user account, and the attacker cannot perform administrative actions with that account.
- Lock down administrative actions as much as you can. If there is no need for your administrators to perform administrative actions from outside the office, then lock down access to those actions to known IP addresses, for example. See [Using Apache to limit access to the Confluence administration interface](#) for guidance.

Further precautions

As another precaution:

- Regularly monitor the above requirements. There are many things that could start out well, but deteriorate over time:
 - A system may start out with just 3 administrators, but over the course of a year this could grow to 30 administrators if no one prevents expansion.
 - Apache administration restrictions may be in place at the start of the year, but when the application server is migrated after a few months, people may forget to apply the rules to the new system.

Again, keep in mind that the above steps may only be a fraction of what could apply to you, depending on your security requirements. Also, keep in mind that none of the above rules can guarantee anything. They just make it harder for an intruder to move quickly.

Brute force login protection

This page explains how to configure or disable FishEye's brute force login protection.

FishEye will protect against brute force login attacks by forcing users to solve a CAPTCHA form after a configurable number of consecutive invalid login attempts. By default, this functionality is enabled, and the number of invalid attempts is set to three.

 Once a user logs in successfully, they will no longer be required to solve the CAPTCHA form.

Configuring brute force login protection

To configure brute force login protection:

1. In the Admin area, click **Authentication** under 'Security Settings'.
2. Scroll down to the 'Security settings' section at the bottom of the screen.
3. Choose 'Use CAPTCHA for login' options. Your changes are applied immediately.

Brute force protection against remote API calls

Login requests by the FishEye remote API libraries are also covered by the brute force protections. After the number of invalid attempts is exceeded (the default is three), then the remote API for that user will be prevented from making further login attempts (as that user will now be required to solve a CAPTCHA form through the web interface in order to log in).

Security aspects

Having the CAPTCHA enabled can leak information whether particular username is registered in the system. We

recommend you disable the CAPTCHA if such leaks present a concern for your organization.

Configuring CAPTCHA

This page contains instructions on enabling and disabling [CAPTCHA](#) testing on the following login-screen features of FishEye and Crucible:

- Public signup.
- Password retrieval.

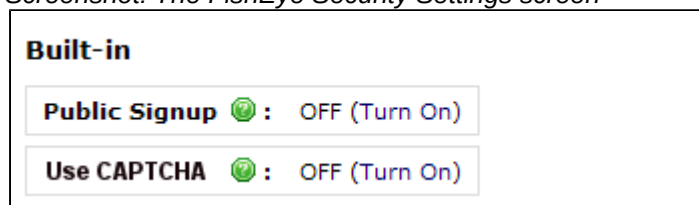
Administrators can enable or disable CAPTCHA testing on the [public signup](#) and [password retrieval](#) screens. CAPTCHA testing is enabled by default.

You may want to turn CAPTCHA off if you are serving FishEye or Crucible behind a firewall.

To do this:

1. Open the FishEye **Admin screen**, then choose '**Security**' from the left navigation bar.
2. The FishEye Security Settings screen opens.
3. Just below '**Public Signup**', the '**Use CAPTCHA**' option can be set to OFF or ON.
4. Select the desired setting by clicking '**Turn (OFF or ON)**'.
5. The setting is immediately changed.

Screenshot: The FishEye Security Settings screen



i To see instructions on configuring CAPTCHA for brute force login protection, see the page on [Brute force login protection](#).

Granting administrator privileges

You can grant Admin privileges to users and groups in FishEye. These privileges exist in addition to the core 'Admin' account.

! Be sparing in granting Admin privileges, as all Admin users have the 'keys to the kingdom'. They can add or remove other's user or group admin rights, and change the password of the core Admin account.

Setting admin privileges for individual users

Log in as the Admin user and click **Administrators**, under 'Security Settings' in the left navigation panel.

Use the >> and << buttons to move a user into or out of the 'Admin Users' column.

! Take care not to remove admin rights from your own account.

Setting admin privileges for groups

Log in as the Admin user and click **Administrators**, under 'Security Settings' in the left navigation panel.

Use the >> and << buttons to move a group into or out of the 'Admin Groups' column.

! Take care not to remove admin rights from your own account.

Configuring public signup

If you enable *public signup* for your FishEye system, visitors can create their own FishEye user accounts using the **Signup** link on the login screen:

Login Required

Username:

Password:

Not a member? [Signup](#) for an account.

Public signup is disabled by default.

To enable public signup:

1. Go to the FishEye [admin area](#).
2. Click **Authentication** in the left navigation column (under 'Security Settings').
3. Click **Turn On**, next to 'Public Signup'.
4. Log out of FishEye and verify that the login screen now contains a **Signup** link.

Authentication Settings ?

Global Permissions

Global Anonymous Access ?	OFF (Turn On)
Crucible Anonymous Access ?	OFF (Turn On)

Built-in

Public Signup ?	OFF (Turn On)
Use CAPTCHA for Signup ?	ON (Turn Off)
User managed mappings	ON (Turn Off)

Username

Force Lowercase Username ?	OFF (Turn On)

Tuning FishEye performance

This page describes a number of ways in which you can optimize the performance of your FishEye installation.

Configurable indexing threads

FishEye is now multi-threaded, allowing you to control the number of threads dedicated to the repository indexing process. See [Configuring indexing](#).

On this page:

- [Configurable indexing threads](#)
- [Java heap size](#)
- [Default number of open files](#)
- [Default number of memory maps](#)
- [Improve browse performance](#)
- [Reduce overall server load](#)
- [Improving initial scan performance](#)

Related pages:

- [Best practices for FishEye configuration](#)

Java heap size

The heap size of the FishEye Java Virtual Machine is controlled by the [FISHEYE_OPTS](#) environment variable. The best heap size to use is dependent on a number of factors including:

- The source code management (SCM) system being used. Subversion scanning typically uses more memory than CVS, for example.
- The complexity of operations in the repository. Processing changesets which affect many files will use more memory.
- The amount of physical RAM in the system. If the Java heap is too large, it may induce swapping which will impact performance.

FishEye will reserve a portion of the available heap for caching of database data. So in general, the more memory you can supply, the better.

For Subversion repositories, it is also possible to reduce FishEye's memory footprint by reducing the [Block Size](#) parameter.

If you do run into 'out of memory' errors, see [Fix out of Memory Errors](#). See also [How FishEye uses memory](#).

Note that when FishEye is run as a Windows service using the Java Service Wrapper, [environment variables](#) are ignored – they must be configured in the wrapper config file instead. See [Running FishEye as a Windows service](#).

Default number of open files

If you have a large number of repositories, we recommend that you increase the default number of files that FishEye is allowed to open. See this knowledge base article for more info: [Subversion Indexer Paused with "Too many open files" Error](#).

Default number of memory maps

If you have a large number of repositories, we recommend that you increase the default number of maps that FishEye is allowed to have. See this knowledge base article for more info: [JVM crashes after FishEye Crucible upgrade - Native memory allocation mmap](#).

Improve browse performance

For users with very large repositories, you may find the rendering of the Subversion tree slow. You can try disabling checking for empty folders by using the [disable-dirtree-empty-checks](#) option in your [FISHEYE_ARGS](#) environment variable.

Reduce overall server load

The [incremental indexing](#) process causes FishEye to poll all repositories at the specified interval to check for new commits, even though there might not be any new information to index. If you have a large number of repositories (> 100), this can lead to:

- A time lag between a commit being made and it appearing in FishEye.
- A high load on the FishEye server, and on the SCM.

[Configuring commit hooks](#) will allow you to set up your SCM so that indexing of a repository is triggered by a commit to the repository itself. This means that FishEye only runs the indexing process when necessary, and allows automatic [polling](#) to be disabled. Commits will appear sooner in FishEye, and the server load will be reduced.

Improving initial scan performance

Background information

When you add a repository, FishEye needs to perform a once-off scan through the repository to build up its initial index and cache. This scan can take some time. Until this scan is complete, you may find that some data is not displayed. As a guide, FishEye should be able to process about 100KB-200KB per second on an averaged-size PC. If FishEye is accessing the repository over the network (e.g. over a NFS mount), then you should expect the initial scan to take longer.

General improvements

You can increase the speed of your scans using the following options:

- If your repository is non-local, set up a local repository mirror on the FishEye server. This will provide a major speed boost for anyone scanning a repository across a network.
- [Exclude](#) unused file types, unused directories and specific large files from FishEye.

Improve update performance during initial scan

One option is break large repositories into multiple smaller repositories. While this technique will not improve the overall initial scan time, it allows for all fully scanned repositories to be updated while the initial scan is still being performed on those remaining.

In FishEye 1.3.4 and later, the initial and incremental scans happen in separate, single threads. So, splitting the repositories will allow incremental scans to run concurrently alongside the initial scans. You may also wish to split projects into separate repositories, since permissions in FishEye are applied on a per-repository basis.

Improving initial scan performance for an SVN repository

svnsync is not an atomic protocol. It generally performs the commit as the svn sync user and then updates the revprops to match the original commit, under some circumstances, this can cause the wrong user to be attributed as the author of a changeset. More information available at [Wrong User Reported When Indexing SVN Mirror Repository](#).

The http/s protocol has the slowest performance during the initial scan. The svn protocol (svn://) is faster and the file protocol (file:///) is the fastest.

Therefore if you find your initial scan takes an extended amount of time (more than a day or two), you should consider switching over from the http/s protocol to the svn or file protocol to define the location of your SVN repository. (Use [svnsync](#) to mirror the repository onto the fisheye server, so that you can access it with the file protocol.)

```
E.g. Switch from
https://example.com/svn/project/
to
svn://example.com/svn/project/
or
file:///home/user/some/location/svn/project
```

In order for SVN protocol to work you need to have set up an [svnserve based server](#).

More information on how to troubleshoot SVN indexing related issues can be found [here](#).

Best practices for FishEye configuration

1. Ensure your FishEye scan performance is as fast as possible.

Use the `file://` protocol for fastest indexing performance. See [Improve FishEye scan performance](#) for more information. (If you cannot install FishEye on the server where Subversion is running, use [svnsync](#) to mirror the repository onto the FishEye server).

2. If your repository is really large, consider starting at a sensible revision

This affects Subversion and Perforce. Do this when defining your [Perforce](#) or [SVN](#) repository.

3. Exclude directories if you don't need them.

For example, not everyone may need to access a developer's personal branch on the repository, so you can exclude it from the repository scan. You may also want to exclude large branches/tags that have been deleted (even though they are deleted in your repository, FishEye will still index them as they once existed). Do this [by using 'Allow' and 'Exclude' Admin settings](#).

4. Consider skipping Perforce Label processing if not important

- Perforce Labels can be slow to process, and thus cause FishEye to index slowly in certain environments.
- Do this by defining the ['Skip Labels' Repository Detail](#).

5. Split your repositories into logical components if you can (For example, by product or project).

- A logical structure will make it simpler to exclude certain branches when they become less relevant to work in progress. This can garner significant performance gains.
- Avoid treating an SCM like a file system — don't alter the structure or move items around without a significant reason for doing so. Make these changes sparingly and as infrequently as possible.
- The more often you make major changes to the structure inside your SCM, the more scanning is required for FishEye to keep track of its status. This especially applies to Subversion, because of its concept of 'cheap copies'. The result is that small changes can be essentially unmeasurable and cause a large amount of re-scanning.

6. Decide on your Subversion tag and branch conventions

Decide what conventions you are going to adopt for your subversion repositories and then stick to them. It's best to stick to one of the standard conventions recommended by Subversion. Best practices for Subversion integration can be found [here](#).

7. Exclude tags and/or branches that you delete and recreate often.

You may have a branch or tag that you delete and recreate often, for example a latest tag which holds the latest release. FishEye will take a long time to index this tag/branch as it needs to index its entire history, which can be very large. It is recommended that you exclude this directory from being indexed. See the [documentation on the 'Excludes' option](#).

8. Avoid using the text `$Log$` in your CVS commit messages.

This is because FishEye does not handle the `$Log` RCS expansion keyword correctly. Some diff results (and line numbers in diffs) may appear incorrect in files where `$Log` is used.

9. Avoid using symbolic links to refer to your FishEye install directory location.

See this existing [Knowledge Base document](#) for more information.

10. Configure your index threads & memory usage to an appropriate level.

See the page [Tuning FishEye](#) for instructions and the related memory guide, [Fix out of Memory Errors](#).

See Also

- [Improve FishEye scan performance](#)
- [Tuning FishEye](#)

Separating FishEye instance data from application data

By default, FishEye will install and run self-contained within the <FishEye install directory>, that is, the directory where you unzip the package. Alternatively, you can create a separate FishEye "instance" directory, referenced by the `FISHEYE_INST` environment variable, outside of the <FishEye install directory>. This may be necessary for practical or administrative reasons. Here are our recommendations for some of these possible scenarios:

Number of Instances	Installation Type	Upgrade Likelihood	Separate FishEye Instance Directory
Single	Production		Highly Recommended
Single	Development	Likely	Highly Recommended
Single	Development	Not Likely	Unnecessary

In order to run multiple instances of FishEye on the same server, it is **mandatory** to establish a separate `FISHEYE_INST` directory.

Number of Instances	Installation Type	Upgrade Likelihood	Separate FishEye Instance Directory
Multiple	Prod. / Dev.	N/A	Mandatory

See the differences between FishEye [folder layouts](#) depending on which installation you choose.

For production installations, we recommend that the FishEye instance directory be [secured against unauthorized access](#).

i As outlined in [Method 3](#) of our [FishEye upgrade guide](#), you can separate your FishEye instance data (to make future upgrades easier) even if initially you chose not to do so:

1. Shut down the FishEye/Crucible instance if it is running.
2. Create a directory outside of the `<FishEye install directory>` path for the FishEye instance data and then set up the `FISHEYE_INST` environment variable.
3. Copy the following items to `/FISHEYE_INST`:
 - `<FishEye install directory>/config.xml`
 - `<FishEye install directory>/var`
 - `<FishEye install directory>/cache`
 - `<FishEye install directory>/data` (optional – if it exists)
4. Start the FishEye/Crucible instance.

Can't See Your Repositories?

If your configuration is not automatically picked up and you cannot see your existing repositories, check your **Administration > Sys-Info** page, where you will see information about `FISHEYE_HOME` (i.e. the install directory) and `FISHEYE_INST`. Check that your `FISHEYE_INST` environment variable is pointing to the correct directory.

Best practices Subversion integration

SVN Client

Atlassian recommends using the [bundled SVNkit client](#) instead of the [native SVN client](#).

Connection Protocol

The `http/s` protocol has the slowest performance during the initial scan. The `svn` protocol (`svn://`) is faster and the `file` protocol (`file:///`) is the fastest.

Therefore if you find your initial scan takes an extended amount of time (more than a day or two), you should consider switching over from the `http/s` protocol to the `svn` or `file` protocol to define the location of your SVN repository. (Use [svnsync](#) to mirror the repository onto the FishEye server, so that you can access it with the `file` protocol.). In order for SVN protocol to work you need to have set up an [svnserve based server](#).

`svnsync` is not an atomic protocol. It generally performs the commit as the `svn sync` user and then updates the revprops to match the original commit, under some circumstances, this can cause the wrong user to be attributed as the author of a changeset. More information available at [Wrong User Reported When Indexing SVN Mirror Repository](#).

On this page:

- [SVN Client](#)
- [Connection Protocol](#)
- [SVN Branch/Tag Structure](#)
- [FishEye Index Structure](#)
- [Limit Indexing Scope](#)
- [Indexing Thread Configuration](#)
- [Further Reading](#)

SVN Branch/Tag Structure

Ensure the SVN branch and tag structure has been selected correctly in FishEye repository configuration. If FishEye does not know which files are tags and branches, it will treat all files as trunk files. This can significantly increase the effective size of your repository and the initial scan time as well as impact runtime performance. Please refer to this guide for [verifying SVN symbolic rules](#).

FishEye Index Structure

FishEye is a caching application. It caches all the repository information. Therefore, big repositories do take time to be indexed.

This index/repository cache is stored in the `var` directory defined in the FishEye configuration, under a subdirectory named `var/cache/name of the repository`. For example, the indexes for the repository FE are under `var/cache/FE`, with one subdirectory for each index:

- Metadata index is stored in `var/cache/FE/idx0`
- Content index is stored in `var/cache/FE/idx1`
- DiffText index is stored in `var/cache/FE/idx2`

Limit Indexing Scope

You can limit the repository content that FishEye will index by:

- Setting a [start revision](#) to start indexing from a particular revision.
- [Exclude](#) certain file types/paths that are not required to be indexed.
- Turn OFF the [store-diff info](#) property. Some of the benefits are:
 - Speed up indexing; FishEye won't have to retrieve and index all diffs.
 - Reduce disk footprint. The diff index files can become quite large, so disabling diffs can lead to significant savings.
 - Reduce disk I/O, particularly write operations. If disk I/O is a performance bottleneck, disabling diff indexing could help.
 - Reduce the load on your SVN server during indexing, but you'll only notice the difference if you've got a lot of commit activity going on.

Indexing Thread Configuration

Check for the [indexing thread configuration](#).

FishEye uses one thread each for incremental and initial indexing for each repository. Three is a recommended thread count. Having more threads will add extra overhead to the server when indexing, and reduce performance.

Further Reading

- [Tuning FishEye Performance](#)
- [Recommended Best Practices](#)

Troubleshooting SVN indexing issues in FishEye and Crucible

This page lists the troubleshooting tips required to investigate some common SVN issues reported while indexing a repository in FishEye and Crucible.

- [Debugging slow indexing](#)
 - [Command line tool](#)
 - [Analyzing debug logs](#)
- [Further resources](#)

Debugging slow indexing

Command line tool

- Monitor the size of your `<FISHEYE_INST>/var/cache/repository_name` folder on disk as this

should be similar to your SVN repository on the server. If there is a huge size mismatch, that is usually an indication of incorrect SVN branch and tag mapping in FishEye.

- If you observe a large size mismatch, you can use the 'svnrules' command line tool to identify missing svn symbolic rules. The tool is shipped with the latest version of FishEye. For more information, see [Verifying SVN symbolic rules](#).

This command may take a while as it runs `svn log -v` against the specified repository. It attempts to analyze all the copy operations in your repository and reports where it is most likely that there are missing svn symbolic rule definitions. Once the tool has finished, check the end of the generated `<Repo Name>-svn-rules.log` file – it contains a summary of the 10 most common folders for which symbolic rules may be missing.

You don't need to stop your current FishEye instance to perform this check. Please note that the tool may report false positives, as sometimes it is genuine activity in the repository to copy some files or folders within the trunk.

Analyzing debug logs

- With debug logging enabled, look for log entries beginning with `[IncrementalPinger<Thread No.> <Repository Name>]`. and `[InitialPinger<Thread No.> <Repository Name>]`. These entries correspond to indexing related tasks currently running in the application and the *Thread No.* corresponds to which thread is being used to complete the task on that repository. Using the timestamp on the log, you can then investigate the last time that FishEye ran an incremental/initial index on a repository.
- Debug logs will also report the exact commands being run on the repository at the time of initial and incremental indexing, for example:

```
2013-04-08 14:05:45,971 DEBUG [IncrementalPinger2 git_tutorial]
fisheye.app LoggingProcessMonitor-onBeforeStart - Starting process: git
fetch /Users/ganand/Git Tutorial/git_tutorial/work/hello/ +refs/*:refs/*

.
.
.
2013-04-08 14:05:46,006 DEBUG [IncrementalPinger2 git_tutorial]
fisheye.app LoggingProcessMonitor-onAfterFinished - Finished process:
git fetch /Users/ganand/Git Tutorial/git_tutorial/work/hello/
+refs/*:refs/* took 33ms
```

In the above example, the "Starting process" and "Finished process" are the messages used to track the start and end of a command and how long did it take to complete - "took 33ms" in this case. This can be helpful in identifying some long running commands or if they do not complete at all when there are no corresponding "Finished process" log entries. You may then run the same commands via the command line directly on your server and compare the results.

- Logs will also lodge the time taken to index a particular repository, for example:

```
2013-04-08 14:06:05,833 INFO [InitialPinger3 git] fisheye.console
BaseRepositoryScanner-ping - [git] starting initial scan of repository

.
.
.
2013-04-08 14:06:11,817 INFO [InitialPinger3 git] fisheye.console
BaseRepositoryScanner-ping - [git] finished initial scan of repository
in 5s
```

- With debug logging enabled, you can also look for the exact command that FishEye is failing at while indexing, for example:


```

2013-04-08 14:28:40,272 DEBUG [InitialPinger3 svn] fisheye.app
SvnThrottledClient-start - Executing ls -r HEAD https://URL/public/@HEAD
2013-04-08 14:28:41,834 WARN [InitialPinger3 svn] fisheye.app
SvnRepositoryTester-getServerRootURL - Unable to get Repository Root URL
for svn:https://URL/public/
2013-04-08 14:28:41,834 DEBUG [InitialPinger3 svn] fisheye.app
SvnRepositoryTester-pingAndValidateAccess - Validating access to
https://URL/public/
2013-04-08 14:28:41,834 DEBUG [InitialPinger3 svn] fisheye.app
SvnThrottledClient-start - Executing propget fisheye.access -r HEAD
https://URL/public/@HEAD
2013-04-08 14:28:43,124 DEBUG [InitialPinger3 svn] fisheye.app
RepositoryStatus-setMessage - Status change [svn]: Could not access
https://URL/public/
2013-04-08 14:28:43,124 ERROR [InitialPinger3 svn] fisheye.app
SvnRepositoryTester-pingAndValidateAccess - Exception getting FishEye
access control
com.cenqua.fisheye.rep.RepositoryClientException:
org.tigris.subversion.javahl.ClientException: svn: E175011: Repository
moved permanently to '/public'; please relocate
svn: E175002: PROPFIND request failed on '/public'

```

In the above example, the command is "svn ls -r HEAD <https://URL/public/@HEAD>". You can run it via the command line to check the result returned from your SVN server.

Further resources

- The FishEye Knowledge base articles for [SVN integration](#) are very useful for identifying the underlying cause.
- You may also use the bundled [Hercules Log Scanner](#) to scan your logs for known issues and get recommendations for resolution accordingly.

How FishEye uses memory

This page describes how FishEye and Crucible use the Java heap when running.

The heap is, of course, used for all FishEye and Crucible operations. The description here, however, is focused mainly on the per-repository aspects of memory usage. The remaining usage of the heap, such as SQL database query caches, Crucible review data, etc is not covered here.

On this page:

- [Per-repository heap components](#)
- [Passivation](#)
- [Database memory allocation](#)
- [String tables](#)
- [Guidelines for heap sizing for a given number of repositories](#)
- [Configuring application heap size](#)
- [Configuring server memory maps](#)

Per-repository heap components

For an active repository, the major components of memory usage for that repository are:

- A database cache. This is an in-memory cache used by FishEye's on-disk database.
- A set of string tables. This is an application level cache of string data used by FishEye. It includes information such as changeset comments, file paths, file names, etc.
- Transient SCM data. This is data fetched from the underlying SCM for the repository. It can include data

needed to create FishEye's database and index. It would also include data fetched from the SCM to satisfy UI requests, such as file revision content. While it is transient, it can still be a significant component of heap usage.

- Transient indexing data. FishEye uses Lucene to index repository metadata and content. Building the Lucene documents and index management uses heap memory.

Passivation

Every active FishEye repository requires an active database connection. When there is a large number of repositories, there may not be sufficient heap available to support having all repository database connections open at once. To support large numbers of repositories running at once, FishEye will transparently "passivate" the repository by closing its database connection and freeing the memory used by both the database cache and the repository's string tables. The repository will be activated, that is, reopen its database connection, when needed. This activation can be triggered by either a UI request or an indexing operation. As a repository activates, FishEye may need to passivate another repository to keep the number of active repositories at a manageable level.

Prior to FishEye 3.5, the repositories were selected for passivation using an LRU algorithm. This makes sense for UI activity but may not have been optimal for indexing activity, where the regular polling behavior means the least recently accessed repository is most likely to be next repository to be accessed. In FishEye 3.5, the UI and indexing needs are balanced by choosing a repository in the middle of the LRU list of repositories.

Database memory allocation

By default FishEye allocates one third of the maximum Java heap size to the repository on-disk database caches. Note that this is for the on-disk B-Tree database and not the SQL relational database, which is managed separately. For example, when running with a 1 GB heap, FishEye will allocate up to 340 MB for database caches.

In FishEye 3.5, the passivation behavior has been changed to take advantage of larger heaps. This includes:

- Raising the minimum per-repository cache size to 5 MB (was previously 1 MB).
- Making the maximum cache size 20 MB.
- Calculating the maximum number of active repositories based on the available heap.

Heap size	Cache allocation	Repository cache size	Max active repositories
1 GB	341 MB	5 MB	68
2 GB	682 MB	5 MB	137
4 GB	1.37 GB	5 MB	273
8 GB	2.7 GB	5 MB	546

In addition, in FishEye 3.5, the level of Java heap garbage collection activity is monitored and additional repositories will be passivated as the time spent in garbage collection increases. This makes the FishEye instance more reactive to increasing UI load.

Prior to FishEye 3.5, the available database cache allocation was shared by the active repositories. If there were 10 repositories in a 1 GB heap, each would be given a 34 MB cache allocation. Also, by default there was a maximum of 50 active repositories before FishEye would begin to passivate repositories. While the 50 repository limit was configurable, it was not automatically changed for FishEye instances with larger heaps. Rather, the increased cache available was used to give the 50 active repositories a larger database cache allocation.

Heap size	Cache allocation	Repository cache size	Max active repositories
1 GB	341 MB	6.8 MB	50
2 GB	682 MB	13.7 MB	50
4 GB	1.37 GB	27 MB	50
8 GB	2.7 GB	55 MB	50

String tables

The string table component of repository memory usage is currently implemented as a weakly referenced cache. This means that as heap memory is used, the JVM garbage collector can remove elements in the cache. The size of the cache is not bounded to a particular size but is bounded by garbage collection activity in the JVM.

The unbounded size of the StringTable can lead to it appearing in heap dumps as a large consumer of heap. This is normal and the string table will decrease in size as the garbage collector (GC) removes entries. In FishEye 3.5, by passivating in response to GC activity, the string tables are flushed more actively when required. This reduces GC activity in the instance as a whole.

Finally, in FishEye 3.5 some string tables that were large in size but little accessed are no longer cached and are now taken directly from the on-disk database.

Guidelines for heap sizing for a given number of repositories

In some respects, repository passivation is like virtual memory in a modern operating system. It allows a FishEye instance to support a larger number of repositories than can be active at once in a given-sized Java heap. Repositories are swapped into the active set on demand and swapped out if not in use.

As the number of repositories configured in a FishEye instance increases beyond the maximum active number supported by the heap, the instance will begin to passivate repositories. Obviously, as a repository database connection needs to be reopened and the in-memory caches need to be repopulated, the onset of passivation comes with a performance impact. As the number of repositories increases, the rate of passivation and activation will increase and the performance impact will also increase. Nevertheless, quite high load factors (ratio of number of repositories to the maximum number of active repositories) can be supported without appreciable performance impact.

Given the cache size limits in FishEye 3.5, the following limits are guidelines to reasonable maximum number of repositories:

Heap size	At max repo cache size	At min cache size / no passivation	Load factor 2	Load factor 5	Load factor 10
1 GB	17	68	136	340	680
2 GB	34	136	272	680	1360
4 GB	68	272	544	1360	2720
8 GB	136	544	1088	2720	5440

What load factors are acceptable? That depends a lot on the nature of the repositories in the instance. If all repositories are equally busy and support a large user base, lower load factors are appropriate. If, on the other hand, some proportion of repositories are accessed sporadically, perhaps a historic codebase, the load factor can be higher as these repositories are not going to activate that frequently.

FishEye 3.5 can, in general, support more repositories with smaller heaps and lower resulting garbage collection load than previous versions. It is also more reactive to memory usage increasing, allowing increasing UI usage to trigger passivation of inactive repositories.

Configuring application heap size

A FishEye or Crucible system administrator can configure the Java VM heap size by setting FISHEYE_OPTS. See [Environment variables](#).

Configuring server memory maps

If you have a large number of repositories, we recommend that you increase the default number of maps that FishEye is allowed to have. See this knowledge base article for more info: [JVM crashes after FishEye Crucible upgrade - Native memory allocation mmap](#).

Contacting Support

Raising a Support Request

There are two ways to raise a support request with Atlassian:

- (Recommended) Raise a Support request directly via the Atlassian [support site](#) and then create a [support zip](#). The advantage of this method is that it includes all the relevant files FishEye/Crucible that Support need. You can also be sure that the support case has been created and includes your logs.
- Complete the [support request form](#) from your FishEye Administration Console. The disadvantage of this method, is that your mail may not be forwarded correctly due to an issue or security restriction on your mail server. For example, the zip of your log files are huge, and your mail server rejects the mail.

Both methods are described in more detail below.

Raise a Support request via Atlassian Support (recommended)

There are two steps.

Step 1: Raise the support request

1. Go to [Atlassian Support System](#).
2. If you do not already have a free Atlassian support account, [create one here](#).
3. Lodge a detailed description of your problem in the new support ticket.
4. Fill in all applicable information about your system, such as application server, database, etc.
5. If FishEye is running:
 - a. Go to the 'Atlassian Support Tools' screen in the FishEye admin area and copy the text of your system information into the ticket.
 - b. Go to **Administration > Server Settings > Debug Logging**, and turn debug logging "ON".
 - c. Reproduce the problem.
 - d. Create a [support zip](#) to attach to the ticket.
6. If your instance does not start up, attach a zip of your FISHEYE_INST/var/logs directory to the support case.
7. Log in to <https://support.atlassian.com> and select **Create New Issue**.
8. Once your ticket is lodged, wait to be notified by email of updates.

Step 2: Create the Support Zip in FishEye/Crucible

 We recommend that you add a [support zip](#) to every interaction with Atlassian Support. The utility will also dump your system information to the logs before zipping them.

You can also use this method to append system information to an existing support request.

1. Log in using an account with admin access.
2. Go to **Administration > Support Tools > Create Support Zip**. Ensure that everything is checked, then click **Create**.
3. Attach the support zip to the support request [you raised](#).

[Create Support Zip](#)[Create Support Request](#)[Log Scanner](#)[Scheduled Log Scan](#)

Create Support Zip

This tool will allow you to create a zip file containing useful information about your instance. No information will be sent to Atlassian.

- ☒ **Application Properties**
Memory statistics, configuration options, and other key information. (required)
 - ☒ **Thread Dumps**
Generate and include a Thread Dumps (3 thread dumps 5 seconds apart). (recommended)
 - ☒ **FishEye/Crucible config.xml File**
Your config.xml file, which contains database settings and other useful information. (recommended)
 - ☒ **FishEye/Crucible Plugin Configuration**
Your FishEye/Crucible plugin configuration files.
 - ☒ **Modified Files**
Include a Copy of All Locally Modified Files. (recommended)
 - ☒ **FishEye/Crucible Application Logs**
Your FishEye/Crucible application logs. (highly recommended)
 - ☒ **FishEye/Crucible fisheye.out Log File**
Your FishEye/Crucible fisheye.out log. (highly recommended)
 - ☒ **FishEye/Crucible Plugin state properties**
FishEye/Crucible Plugin state properties. (recommended)
 - ☒ **Limit File Sizes?**
Limit the size of files included in your support zip to no more than 25Mb.
- [Create](#)

Raise a Support request via the FishEye Administration Console

i Ensure that **SMTP email** is set up on your FishEye instance and your mail server allows zip files.

The advantage of this method is that it is convenient, however the disadvantage is that your mail may not be forwarded correctly due to an issue (e.g. zip file too large) or security restriction on your mail server.

You can also use this method to append system information to an existing support ticket.

On the left navigation bar, click **Support Tools** and then the **Create Support Request** tab. Fill out the web form which will automatically send an email to Atlassian Support, attaching your FishEye logs and configuration file (if you wish).

i This functionality requires that the FishEye web server is already set up and capable of sending email.

Create Support Zip
Create Support Request
Log Scanner
Scheduled Log Scan

Create Support Request

To: [Atlassian Support Team](#)

Contact Email:

Summary:

Priority:

Description:

Support Data to Attach

Attaching logs and configuration information to your Support Request helps our engineers give you a faster and more complete response. Usernames and passwords are removed from your configuration files automatically (usernames will still appear in log files).

- ☒ **Application Properties**
Memory statistics, configuration options, and other key information. (required)
- ☒ **Thread Dumps**
Generate and include a Thread Dumps (3 thread dumps 5 seconds apart). (recommended)
- ☒ **FishEye/Crucible config.xml File**
Your config.xml file, which contains database settings and other useful information. (recommended)
- ☒ **FishEye/Crucible Plugin Configuration**
Your FishEye/Crucible plugin configuration files.
- ☒ **Modified Files**
Include a Copy of All Locally Modified Files. (recommended)
- ☒ **FishEye/Crucible Application Logs**
Your FishEye/Crucible application logs. (highly recommended)
- ☒ **FishEye/Crucible fisheye.out Log File**
FishEye/Crucible Plugin state properties. (recommended)
- ☒ **Limit File Sizes?**
Limit the size of files included in your support zip to no more than 25Mb.

Fields in the Support Request form

On the Support Request form there are a number of fields to fill out and options to select.

Subject	Enter a one line summary of the problem.
Priority	Choose from Low, Medium, High or Critical.
Description	Type a detailed description of the problem you are trying to solve.
Existing Support Request	Leave this field blank if you are creating a new support request. If you have an existing support request open at support.atlassian.com , enter the issue key here. Doing this will append this new request to the existing request.
Contact Name	Your contact name.
Contact Number	Your contact number.
Attach FishEye Logs	Tick this box to send Atlassian your FishEye log file.
Attach Config File	Check this box to send Atlassian your FishEye configuration file.
Support Entitlement Number (SEN)	Paste your Support Entitlement Number. See Finding your FishEye or Crucible Support Entitlement Number (SEN) .

When you have filled out the required fields, click **Send Support Request** to finish.

⚠ Note that the form may take several minutes to fully submit, as it takes some time to export and process the log file data.

Finding your FishEye or Crucible Support Entitlement Number (SEN)

There are three ways to find your Support Entitlement Number (SEN):

▼ Method 1: Check in the FishEye Administration Interface

Select Administration >> SysInfo/Support. The SEN is shown:

The screenshot shows the 'MyAtlassian' interface with the 'Licenses' tab selected. A callout box points to the 'Support Entitlement Number' field in the license details for 'Confluence 500 Users: Commercial License'. The SEN is displayed as 'SEN: [redacted]'. Other details include the license name 'Atlassian', support expiration date '10 Apr 2011', and a 'Request Support' link. The interface also shows a table of license history and contact information for technical and billing contacts.

Product	Name	Support Expires	Support
Confluence 2 Users: Personal Server	[redacted]	02 Jun 2009	
Confluence 500 Users: Commercial License	Atlassian	10 Apr 2011	Request Support

License Details for Confluence 500 Users: Commercial License

Name: Atlassian (edit)
 SEN: SEN: [redacted]
 Server ID: [redacted] (edit)

Technical Contact: [redacted]
 Email: [redacted] (Add)

Billing Contact: [redacted]
 Email: [redacted] (Add)

Licensed To: Atlassian
 License: [redacted]

This license key is compatible with Confluence 3 or above. If you are using an older version of Confluence, you can [downgrade your license key](#).

Actions: [Renew](#) | [Upgrade](#) | [Download Source](#) | [View Developer License](#)

History:

10 Apr 2008	Initial Purchase
10 Apr 2009	Maintenance Renewal
01 Jun 2009	Converted to Confluence 500 Users: Commercial License
10 Apr 2010	Maintenance Renewal

▼ Method 3: Atlassian Invoice

Your Support Entitlement Number (SEN) appears on the third page of your Atlassian Invoice.

See [Finding Your Support Entitlement Number](#) in the support space for more general information about how Atlassian Support uses this number.

FishEye FAQ

FishEye FAQ

Answers to frequently asked questions about configuring and using FishEye.

- Top Evaluator Questions
 - How do I fix problems with indexing my repository?
 - How do I setup JIRA integration?
 - How do I setup LDAP or external user management?
 - How do I speed up slow CVS updates?
 - How do I start FishEye as a Windows service?
 - How do I view changesets and diffs?
 - How is FishEye licensed?
 - What kind of search capabilities does FishEye have?
 - What programming languages are supported?
- General FAQ
 - About database encoding
 - About the Lines of Code Metric
 - Cannot View Lines of Code Information in FishEye
 - Finding your Server ID
 - How Do I Archive a Branch within Perforce
 - How do I Avoid Long Reindex Times When I Upgrade
 - Mercurial Known Issues
 - Ordering of Branches Important When Visualizing Git Changesets
 - Permanent authentication for Git repositories over HTTP(S)
 - Perforce Changesets and Branches
 - What SCM systems are supported by FishEye?
 - Automating Administrative Actions in Fisheye
 - How are indexing requests handled when they are triggered via commit hook
- Installation & Configuration FAQ
 - Can I deploy FishEye or Crucible as a WAR?
 - Does Fisheye support SSL (HTTPS)?
 - Improve FishEye scan performance
 - Migrating FishEye Between Servers
 - Setting up a CVS mirror with rsync
 - What are the FishEye System Requirements?
 - How to reset the Administration Page password in FishEye or Crucible
 - How Do I Configure an Outbound Proxy Server for FishEye
 - How to remove Crucible from FishEye 2.x or later
 - How to run Fisheye or Crucible on startup on Mac OS X
- Licensing FAQ
 - Are anonymous users counted towards FishEye's license limits?
 - Restrictions on FishEye Starter Licenses
 - Updating your FishEye license
 - Git or Hg Repository exceeds number of allowed committers
- Example EyeQL Queries
 - How do find changes made to a branch after a given tag?
 - How do I filter results?
 - How do I find changes between two versions, showing separate histories?
 - How do I find changes made between two version numbers?
 - How do I find commits without comments?
 - How do I find files on a branch, excluding deleted files?
 - How do I find files removed from a given branch?
 - How do I find revisions made by one author between versions?
 - How do I select the most recent revisions in a given branch?
 - How do I show all changesets which do not have reviews?
- Integration FAQ
 - How do I disable the Source (FishEye) tab panel for non-code projects?
 - How do I enable debug logging for the JIRA FishEye plugin?
 - How do I uninstall the JIRA FishEye plugin?
 - How is the Reviews (Crucible) tab panel for the JIRA FishEye Plugin populated?
 - What do I do if I discover a bug with the JIRA FishEye plugin?

- [Subversion FAQ](#)
 - [Configuring Start Revision based on date](#)
 - [Errors 'SEVERE assert' or 'Checksum mismatch'](#)
 - [FishEye fails to connect to the Subversion repository after a short time of successful operation.](#)
 - [How can FishEye help with merging of branches in Subversion?](#)
 - [Subversion Changeset Parents and Branches](#)
 - [SVN Authentication Issues](#)
 - [What are Subversion root and tag branches?](#)
 - [Why do I need to describe the branch and tag structure for Subversion repositories?](#)
 - [Why don't all my tags show up in FishEye?](#)
 - [About merges in Subversion](#)
- [CVS FAQ](#)
 - [How does FishEye calculate CVS changesets?](#)
 - [How is changeset ancestry implemented for CVS?](#)
- [Support Policies](#)
 - [Bug Fixing Policy](#)
 - [New Features Policy](#)
 - [Security Bugfix Policy](#)
- [Troubleshooting](#)
 - [After I commit a change to my CVS repository, it takes a long time before it appears in FishEye.](#)
 - [FishEye freezes unexpectedly](#)
 - [I have installed FishEye, and the initial scan is taking a long time. Is this normal?](#)
 - [I have installed FishEye, but there is no data in the Changelog.](#)
 - [Initial scan and page loads are slow on Subversion](#)
 - [JIRA Integration Issues](#)
 - [Manually Generating a Thread Dump](#)
 - [Message 'org.tigris.subversion.javahl.ClientException svn Java heap space'](#)
 - [Problems with very long comments and MySQL migration](#)
 - [URLs with encoded slashes don't work, especially in Author constraints](#)
 - [Manually Generating a Thread Dump \(Draft\)](#)
 - [All Repositories fail - Could not execute query \(MySQL database\)](#)
- [FishEye Developer FAQ](#)
- [Contributing to the FishEye Documentation](#)
 - [FishEye Documentation in Other Languages](#)
- [FishEye Resources](#)
- [Glossary](#)
- [Collecting analytics for FishEye](#)

Do you have a question, or need help with FishEye? Please [create a support request](#).

General FAQ

FishEye General FAQ

- [About database encoding](#)
- [About the Lines of Code Metric](#)
- [Cannot View Lines of Code Information in FishEye](#)
- [Finding your Server ID](#)
- [How Do I Archive a Branch within Perforce](#)
- [How do I Avoid Long Reindex Times When I Upgrade](#) — If reindexing your repository takes longer than you can allow, you can use a temporary copy of your repository and FishEye instance to reduce downtime during the reindexing process.
- [Mercurial Known Issues](#)
- [Ordering of Branches Important When Visualizing Git Changesets](#)
- [Permanent authentication for Git repositories over HTTP\(S\)](#)
- [Perforce Changesets and Branches](#)
- [What SCM systems are supported by FishEye?](#)
- [Automating Administrative Actions in FishEye](#)
- [How are indexing requests handled when they are triggered via commit hook](#)

About database encoding

It is possible to have files in your repository whose names differ only in case, e.g. Foo.java and foo.java. Hence, your database will need to use rules for comparing string values which recognize that upper and lower case letters are different, that is, the database should use 'case sensitive collation'.

If your database was originally configured to use case-insensitive and/or non-UTF8 collation, FishEye will display the following message at the bottom of your screen:

"Your database is not using a case sensitive UTF8 encoding for character fields."

The following sections provide instructions for changing your database collation for each database type supported by FishEye and Crucible.

On this page:

- [MySQL](#)
- [Oracle](#)
- [PostgreSQL](#)
- [SQL Server](#)

Related pages:

- [Migrating to an external database](#)

MySQL

Please take a backup of your database before changing its collation.

To change your collation to utf8_bin you need to change your database's default collation, but as this only affects newly created tables you will also need to change the collation on the table for which case sensitivity is critical.

Change your database's collation

Use the **ALTER DATABASE** command, as follows:

```
alter database character set utf8 collate utf8_bin;
```

Change collation for the CRU_STORED_PATH table

Use the `ALTER TABLE` command, as follows:

```
alter table cru_stored_path convert to character set utf8 collate
utf8_bin;
```

Oracle

Oracle collation encoding must be configured when installing the database server. It cannot be configured on a per database level. When installing Oracle, you should select the AL32UTF8 encoding.

PostgreSQL

Please take a backup of your database before changing its collation.

If you have created your PostgreSQL database with the incorrect encoding, you will need to dump your database, drop it, create a new database with the correct encoding and reload your data.

You can do this using the [standard database migration procedure](#) – instead of migrating from HSQLDB to PostgreSQL, you migrate from a PostgreSQL database with the incorrect encoding to one created with the correct encoding.

SQL Server

Please take a backup of your database before changing its collation.

Unfortunately, changing the database collation for an existing SQL Server database (even using the `ALTER DATABASE ... COLLATE...` statement) does not change the collation for existing objects stored in the database. See <http://blogs.msdn.com/b/qingsongyao/archive/2011/04/04/do-not-alter-database-collation-in-your-server.aspx> for an explanation of this.

The recommended route for changing the collation for SQL Server is to migrate to a new database that has the correct collation configuration. You can do this using the [standard database migration procedure](#) – instead of migrating from HSQLDB to SQL Server, you migrate from a SQL Server database with the incorrect collation to one created with the correct collation.

The correct collation to use when you create the new SQL Server database is `Latin1_General_CS_AS`.

About the Lines of Code Metric

This page contains information about the Lines of Code metric and how it is processed and represented by FishEye.

On this page:

- [Definition](#)
- [Disadvantages](#)
- [LOC in FishEye](#)
- [User-Specific LOC](#)

Definition

Lines of Code or LOC (also known as Source Lines of Code - SLOC) is a quantitative measurement in computer programming for files that contains code from a computer programming language, in text form. The number of lines indicates the size of a given file and gives some indication of the work involved.

LOC is literally the count of the number of lines of text in a file or directory. In FishEye, blank lines and comment

lines are counted toward the total lines of code.

LOC for a file/directory is the total number of lines in the relevant files, while LOC for an author is the number of lines **blamed on that author**. Neither of these should ever be less than zero. However, the **change** in LOC over a period of time can be negative if there was a net reduction in the LOC over the period.

Disadvantages

While it can be useful, LOC has some [well documented disadvantages](#). Keep these disadvantages and limitations in mind when using LOC in your work environment.

In addition, the nature of branching in [SCM](#) applications means that calculating a LOC value for a whole project is not possible. A naive summation of the LOC of all the branches will give a meaningless number that jumps every time a branch is copied to create a new branch. Thus, in FishEye we usually look at the LOC of the trunk, unless we can infer from the context that another branch is more appropriate.

LOC in FishEye

FishEye calculates the LOC for the trunk only. For SVN repositories it can calculate LOC for a branch if it is "[tricked](#)" to see the branch as part of the trunk. FishEye also calculates the LOC for each user, unless that facility is turned off in the repository (see [Store Diff Info](#)). The LOC count will include all files except those identified by the SCM as binary.

FishEye presents LOC data as charts of the change in LOC over time, and as informational statistics in various places:

- **Chart pages**
The best way to explore the evolution of LOC in your project is the [LOC chart report](#) where you can easily filter the LOC by branch, author, file extension and date range. Here you can investigate what caused a particular spike in the LOC charts, or find the user whom has the most lines of code blamed on them and how this has changed over time.
- **Repository-specific activity pages**
These show trunk LOC statistics for the repository, limited to the directory being viewed and its subdirectories. The LOC charts show the LOC for the directory, using trunk LOC unless the directory can be identified as a branch.
- **User pages**
Here, the statistics pane in the sidebar shows the trunk LOC blamed on the user for the all repositories that have user-specific LOC enabled. The chart shows the trunk LOC from all the repositories that the user has contributed to.
- **The global User List page**
This shows the trunk LOC for all users from the repositories that have user-specific LOC enabled. Repository-specific user lists (in repositories that have user-specific LOC enabled) show the trunk LOC for the users and committers, limited to the directory being viewed and its subdirectories.
- **Project pages**
This shows a chart of the LOC for all associated repository paths, and statistics include the trunk LOC for those paths.

User-Specific LOC

The evolution of user-specific LOC over subsequent commits can appear at first glance to be counter-intuitive. It is important to keep in mind that the LOC for a given user is the number of lines in the repository that were last changed by them (as calculated by FishEye).

A couple of simple examples:

- Alice adds a files with 30 lines to the SCM. Her LOC for this file is now 30. She then edits the file, deletes 10 lines and adds 20 (+20 -10). Her LOC is now 40, as is the LOC of the file.
- Alice adds a files with 30 lines to the SCM. Her LOC for this file is now 30. Now Bob edits the file, deletes 10 lines and adds 20 (+20 -10). Alice now has LOC of 20, because Bob deleted 10 lines that were blamed on her, and Bob has LOC of 20, from the 20 lines he added. The total LOC is still 40.

A user can have LOC on a branch that they have never committed on, *if something that has been blamed on them is copied*. For example, a developer may have never committed to a particular branch, but FishEye may still report a lot of LOC for them in that area.

One current limitation of FishEye's user-specific LOC calculation is the handling of merging. For example, if a file has been changed on both trunk and branch, and the changes made on the branch are merged to trunk, the changes made on branch will generally be blamed on the person who did the merge; **not** the person who made the change.

Cannot View Lines of Code Information in FishEye

Symptoms

The LOC (Lines of Code) information in FishEye cannot be seen, for example in [charts](#) or when [viewing the statistics for a user](#).

 See [About the Lines of Code Metric](#) for more information about the usage of the LOC (Lines of Code) metric in FishEye.

Cause

There are four possible causes:

- LOC data will not be shown for users if the [Store diff info](#) setting is disabled. If a page is being viewed in FishEye that relates to a particular user or committer, and the Store Diff Info setting is disabled, no LOC information for the user will be visible.
- LOC data is currently not supported for [Mercurial](#) repositories.
- LOC data is currently not supported for [Git](#) repositories.
- The SVN repository is indexing branches only.

Resolution

Cannot view LOC information for specific users or committers:

- Enable the [Store Diff Info](#) setting for the repository in Administration > Repository Settings > Repositories, click on the repository name, and then "SCM Details". A full re-index needs to be performed on the repository after enabling this setting for FishEye to collect the diff information for all revisions in the repository. Please note that the Store Diff Info setting is always enabled for CVS repositories.

The SVN repository is indexing branches only:

- FishEye can calculate LOC for a branch if it is ["tricked" to see the branch as part of the trunk](#).

Finding your Server ID

Your Server ID can be found in the FishEye administration area.

To find your Server ID:

1. Navigate to FishEye's administration area.
2. Click **System Info** (under 'System Settings').

The Server ID for your FishEye server is displayed in the 'License' section.

The Server ID should match the one set for your license. You can check this at <http://my.atlassian.com>.

How Do I Archive a Branch within Perforce

In SVN, a branch exists as a separate directory. However in Perforce, files are given a label to identify them as belonging to the branch. Thus it may not be possible to download the branch as a tarball via FishEye.

You may be able to download the branch as a tarball, depending on your structure:

If it is not a single folder, then it is not possible to download the tarball in your perforce repository.

1. In FishEye, navigate to your perform repository.
2. In the Constraint section on the left, select the branch. This will return the directories that belong to that branch.
3. If it is one single folder, download the tarball of it. Under constraint and sub directories, there is a panel tarball giving options on how to download the directory.

How do I Avoid Long Reindex Times When I Upgrade

Mitigating lengthy reindex times

If reindexing your repository takes longer than you can allow, you can use a temporary copy of your repository and FishEye instance to reduce downtime during the reindexing process.

Most upgrades (even major ones) do not require a reindex. If a reindex is required, this will always be explicitly mentioned in the Upgrade Guide for that release.

On this page:

- [Mitigating lengthy reindex times](#)
- [Reindexing with a temporary copy of your FishEye instance](#)
 - [How to make a temporary copy of your FishEye instance](#)
 - [How to make a temporary copy of your repository](#)
 - [How to reindex a single repository on a test server](#)
- [Upgrading your cross-repository index using a temporary staging server](#)

Reindexing with a temporary copy of your FishEye instance

This section describes how to perform a full reindex of a particular repository. Note that, depending on the repository size, the reindex could take up to several days.

To reindex a temporary copy of your FishEye instance:

1. Make a copy of your FishEye instance to another server. See 'How to make a temporary copy of your FishEye instance' below for instructions.
2. Upgrade the temporary FishEye, then start it up, connected to your repository. It will automatically begin the scanning process.
 -  If you are concerned about the repository being overloaded by the scanning process, you can make a copy of that as well. See 'How to make a temporary copy of your repository' below for instructions. If you do that, you must edit the `config.xml` of your temporary FishEye instance to point to your temporary repository.
3. The copied instance will run its course without affecting your production instance.
 - a. Shutdown both your servers completely.
 - b. Make a backup of your `FISHEYE_INST` directory.
 - c. Replace the `FISHEYE_INST/var/cache` directory on live FishEye with the `FISHEYE_INST/var/cache` from your test server.
 - d. Download the latest FishEye/Crucible from [Atlassian downloads](#).
 - e. Follow the instructions in the [Upgrade Guide](#) to upgrade to the new version.
4. The scan of the temporary FishEye instance (and repository, if you copied that also) is complete. You're now free to delete the temporary copy(s).

How to make a temporary copy of your FishEye instance

To make a copy of your FishEye instance, follow the instructions for [Migrating FishEye Between Servers](#).

How to make a temporary copy of your repository

To make a copy of your repository use [rsync](#) (for CVS repositories in the Linux environment) or [svnsync](#) documentation (for Subversion only).

How to reindex a single repository on a test server

If you need to reindex your repository on your production system but don't want to burden your production server, carry out the following steps:

1. Install another instance of FishEye on a test server (the same FishEye version as the one you are using).
2. Add a repository to FishEye with the exact same name and details as that referenced by the production server.
3. Let it finish indexing. Go to **Administration > View Repository List > Stop** (shown next to the name of your repository) and disable on both production and test.
4. Copy the `FISHEYE_INST/var/cache/REPO` directory from the test server over the `FISHEYE_INST/var/cache/REPO` directory on the production FishEye.
5. Trigger a review revision data re-index: **Administration > Repository > Maintenance > Review-Revision Data Index**.

 For this procedure, neither server needs to be shut down.

Upgrading your cross-repository index using a temporary staging server

This section describes how to upgrade the cross-repository index for selected repositories. Note that, depending on the repository size, the reindex will typically finish in a few hours, but should never take longer than a few days.

In this procedure it is assumed that you have a production server (referred to as *PROD* in these instructions) that is running a FishEye version earlier than 3.1, and a separate staging server (*STAGING*) that will be used to perform the cross-repository index upgrade offline.

1. Make a live backup of the *PROD* server with the following options:
 - a. Repository and application caches
 - b. SQL database

You can do this either from the FishEye Admin area (go to **Administration > System Settings > Backup**), or from a command line, for example:

```
$ ./bin/fisheyectl.sh backup -f ~/Documents/backup.zip --no-uploads
--no-templates --no-plugins --cache --no-ao
```

2. Install FishEye 3.1, or a later version, on the *STAGING* server.
3. Restore the backup of *PROD* to the *STAGING* server.
4. Start FishEye on the *STAGING* server. Note that:
 - The cross-repository index upgrade will start automatically on the *STAGING* server. If you want to perform the cross-repository index upgrade for selected repositories only, it is safe to remove unwanted repositories from the *STAGING* server now, either by going to **Administration > Repositories**, or by using REST endpoints (see below).
 - The *STAGING* server doesn't need to have access to configured SCM's as the cross-repository upgrade task does not interact with them.
 - You may want to disable polling on the *STAGING* server. You can either go to **Administration > Repository Settings > Defaults > Updater** to disable polling for all repositories (although this will *not* affect particular repositories that have been configured to ignore default settings), or go to **Administration > Repository Settings > Repositories > Repository X > Updates** to disable polling for just *Repository X*. Disabling polling is not required, but will avoid logging errors to the FishEye log file if the SCMs are not accessible from the *STAGING* server.
5. Wait for the cross-repository index upgrade to finish on the *STAGING* server. Check by going to **Administration > Repositories**.
6. Stop the *STAGING* server.
7. Make a full backup of the *PROD* server and then stop it.

8. Install the same version of FishEye on the *PROD* server as used on the *STAGING* server (as in step 2 above).
9. Delete the following FishEye indexes on the *PROD* server and replace them with the equivalent caches from the *STAGING* server. You can choose your preferred option to copy files between machines using ssh/scp/rsync, possibly combined with tar/zip tools. The example below shows how the scp command could be used:

```
ssh PROD
cd FISHEYE_INST # replace FISHEYE_INST with the location of your
FISHEYE_INST folder
rm -rf cache/globalfe
rm -rf var/cache/repoX # repeat for each repository repoX that was
upgraded on STAGING server
scp -r STAGING:STAGING_FISHEYE_INST/cache/globalfe cache/
scp -r STAGING:STAGING_FISHEYE_INST/var/cache/repoX var/cache/ #
repeat for each repository repoX that was upgraded on STAGING
server
```

10. Start the *PROD* server.
11. All the changesets that were added to SCMs after backing up the *PROD* server in step 1 will now be indexed on the *PROD* server.

The only drawback with this procedure is that changeset comments added for changesets in *PROD* after step 1 will not get indexed, so they will not appear in the activity stream. There is no easy way to reindex them, apart from fully reindexing each affected repository, which is what this procedure is intended to avoid. A new REST endpoint could be implemented to address this (see

✓ [FECRU-3764](#) - new REST endpoint to reindex changeset comments for a repo CLOSED).

Note: the following REST endpoint could be used to force a cross-repository index upgrade for a selected repository: [/rest-service-fecru/admin/repositories-v1/repoX/reindex-search](#). There should be no need to use this, but it may be useful if something goes wrong.

Mercurial Known Issues

- [CRUC-3474](#): If a file is removed and then another file is copied or moved over the same file within one commit, the ancestor revision is miscalculated and can result in errors in "diff to previous".
- [CRUC-3470](#): Permission changes (and prop changes in repos converted from svn) may result in revisions that have no ancestors - subsequent changes will consider it's parent revision to be their parent revision.
- [CRUC-3468](#): Scanning repositories converted from svn (especially using hgsubversion) can result in commits that take a long time to scan (due to the changes produced by merges from other branches).

Ordering of Branches Important When Visualizing Git Changesets

FishEye 2.6 introduced the [repository commit graph](#). The commit graph allows you to visualize changesets in their branches by showing them in configurable branch "swimlanes". One of the ways in which you can configure the commit graph is by reordering the swimlanes. Reordering swimlanes is useful for non-Git repositories, if you want to show branches in a certain order. However, ordering swimlanes is vital for Git repositories, as it is the only way of determining which branch a commit is displayed in, when a commit belongs to multiple branches.

Git Branches and Changesets in FishEye

Before considering how Git repositories are visualized in the commit graph, it is important to understand how FishEye relates Git changesets to branches.

In FishEye 2.6 and later, FishEye considers the ancestry of a Git changeset when determining which branch it is a part of. Branches can effectively be considered as pointers to changesets. Hence, merging and branching can change the branches that a changeset is considered part of.

For example, if a branch 'fisheye-2.6' is merged back to the 'master' branch, then all changesets that were seen as part of the 'fisheye-2.6' branch only will also be considered to be part of the 'master' (e.g. the changeset will be seen as part of 'master' and 'fisheye-2.6' in the activity stream).

Viewing Git Changesets and Branches in the Commit Graph

The previous section describes how a changeset can be associated with multiple branches, due to its ancestry. Instead of showing the changeset in every branch swimlane on the commit graph, FishEye represents these changesets as described below.

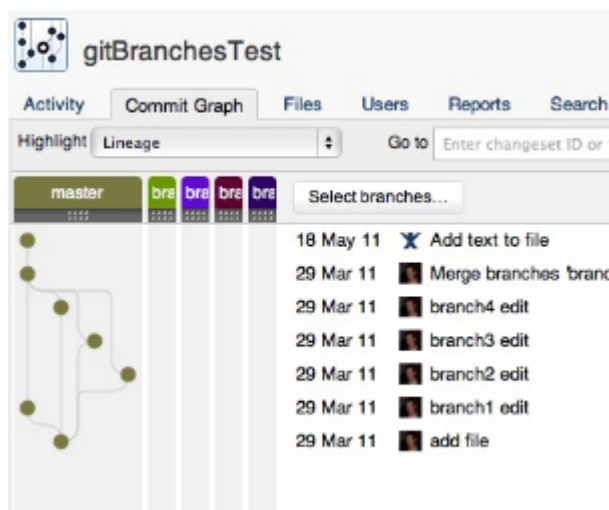
When you view the commit graph for a Git repository, FishEye works from the leftmost swimlane to the right doing the following:

- For each swimlane, FishEye checks if the commit is in that branch. If the commit is in the branch, a dot is shown representing the commit.
- If the commit is not in the branch, the dot for the commit is moved to the next column on the right.

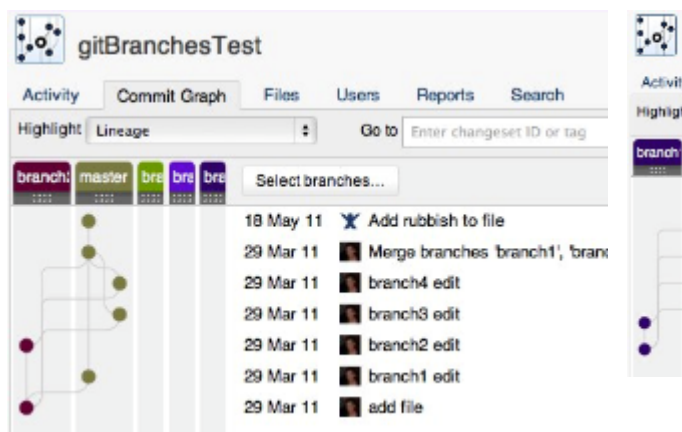
For example, if the 'master' swimlane is to the left of another swimlane, e.g. 'fisheye-2.6' branch, there will be no changesets shown in the 'fisheye-2.6' swimlane, as all the commits will be picked up in the 'master' swimlane. However, if you move the 'fisheye-2.6' swimlane to the left of the 'master' swimlane, it will pick up all of the FishEye 2.6 commits.

This allows you to visually isolate changesets in the desired branches by reordering the swimlanes. For example, if you want to see the lineage of a branch, 'fisheye-2.6', but not 'fisheye-2.5' after both branches have previously been merged to 'master', you could arrange your swimlanes to 'fisheye-2.6', 'master', 'fisheye-2.5' from left to right. You will be able to see the 'fisheye-2.6' changesets and where the merge back to 'master' was made. The 'fisheye-2.5' changesets will just be seen as part of the 'master' branch.

Screenshots below: Example of how ordering swimlanes affects the branches that changesets are displayed on (click to view full-size images)



1



2

Permanent authentication for Git repositories over HTTP(S)

Currently, FishEye only supports HTTP or HTTPS for pushing and pulling from Git repositories. Git has no method of caching the user's credentials, so you need to re-enter them each time you perform a clone, push or pull.

Fortunately, there is a mechanism that allows you to specify which credentials to use for which server: the `.netrc` file.

Warning!

Git uses a utility called `cURL` under the covers, which respects the use of the `.netrc` file. Be aware that other applications that use `curl` to make requests to servers defined in your `.netrc` file will also now be authenticated using these credentials. Also, this method of authentication is potentially unsuitable if you are accessing your FishEye server via a proxy, as all `curl` requests that target a path on that proxy server will be authenticated using your `.netrc` credentials.

Warning!

cURL will not match the machine name in your `.netrc` if it has a username in it, so make sure you edit your `.git/config` file in the root of your clone of the repository and remove the user and '@' part from any clone url's (url fields) that look like `https://user@machine.domain.com/...` so instead they look like `http://machine.domain.com/...`

Linux or OSX

1. Create a file called `.netrc` in your home directory (`~/ .netrc`). Unfortunately, the syntax requires you to store your passwords in plain text - so make sure you modify the file permissions to make it readable only to you.
2. Add credentials to the file for the server or servers you want to store credentials for, using the format below. You may use either IP addresses or hostnames, and you **do not** need to specify a port number, even if you're running FishEye on a non-standard port.

```
machine fisheye1.mycompany.com
login myusername
password mypassword
machine fisheye2.mycompany.com
login myotherusername
password myotherpassword
```

3. And that's it! Subsequent `git clone`, `git pull` and `git push` requests will be authenticated using the credentials specified in this file.

Windows

1. Create a text file called `_netrc` in your home directory (e.g. `c:\users\kannonboy\_netrc`). Curl has problems resolving your home directory if it contains spaces in its path (e.g. `c:\Documents and Settings\kannonboy`). However, you can update your `%HOME%` environment variable to point to any old directory, so create your `_netrc` in a directory with no spaces in it (for example `c:\curl-auth\`) then set your `%HOME%` environment variable to point to the newly created directory.
2. Add credentials to the file for the server or servers you want to store credentials for, using the format from the **Linux or OSX** section above.

Perforce Changesets and Branches

Why does FishEye say this changeset is on more than one branch? Why does that changeset have multiple parents?

Perforce allows a single changeset to include files on multiple branches, so FishEye marks those changesets as being on all of the branches involved.

When a changeset is on multiple branches FishEye may consider it to have multiple parents from the different branches.

FishEye does not track merges in Perforce, so merges are not shown in the graph.

Changeset branches and parents are only annotations at the changeset level --- the individual file revisions will each only have a single branch and at most one parent.

What SCM systems are supported by FishEye?

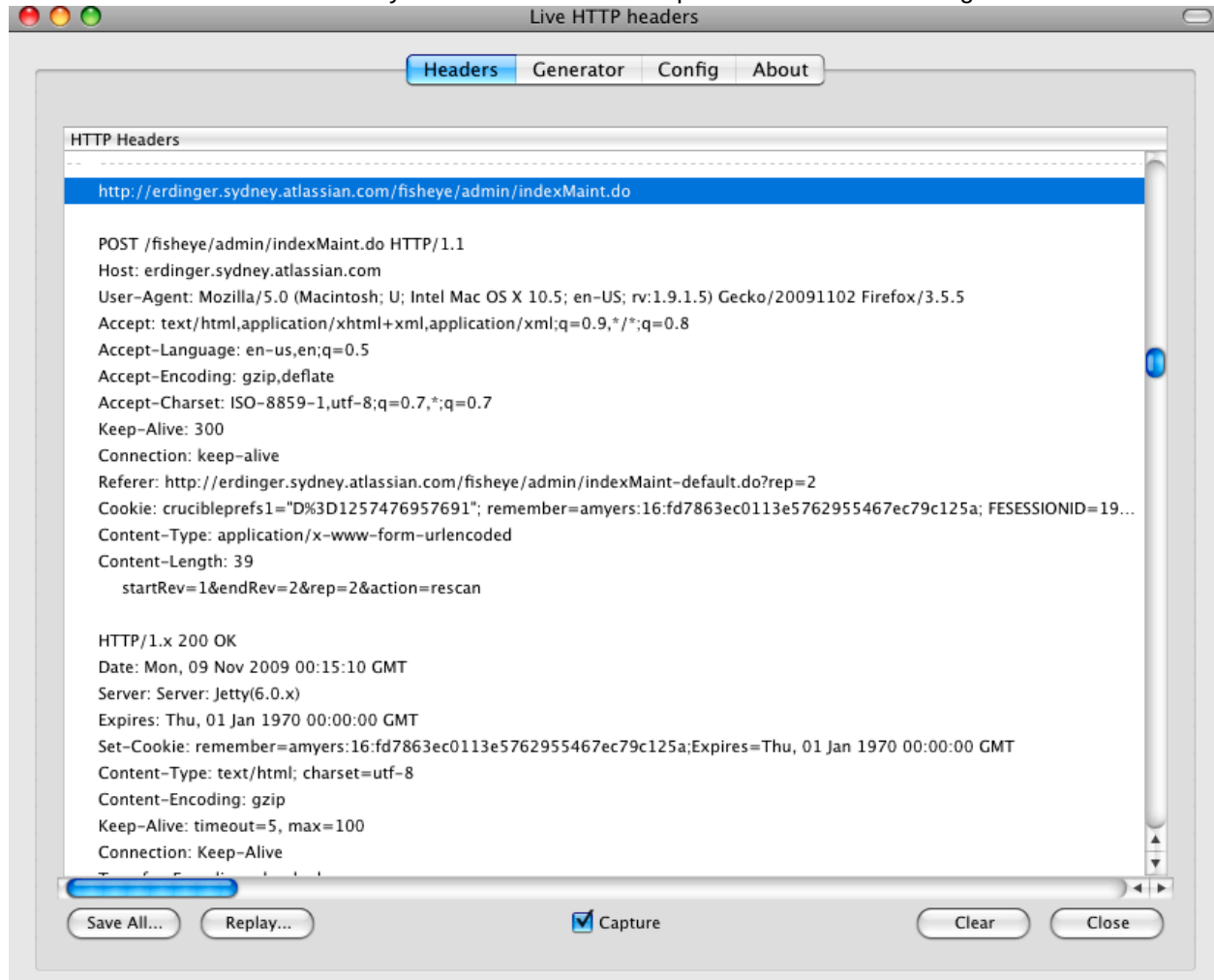
To see the list of SCM systems that is supported by FishEye, see [Supported platforms](#).

Automating Administrative Actions in Fisheye

With some command line scripting and a tool like [wget](#), and [Live HTTP Headers](#) for firefox you can automate actions. In this example, Fisheye will automatically rescan revision properties of an SVN when the commit message is updated to reference a new JIRA issue.

1. Enable live HTTP headers in firefox, then perform the action you want to perform automatically via the Fisheye Administration UI.

2. In the live HTTP headers window you should see some output similar to the following:



3. The important parts are the URL I've highlighted above (<http://erdinger.sydney.atlassian.com/fisheye/admin/indexMaint.do>) and any GET/POST parameters (startRev=0&endRev=58&rep=2&action=rescan)
4. Now we can construct a script with wget to automate this:

```
wget --keep-session-cookies --save-cookies cookie.txt
http://erdinger.sydney.atlassian.com/fisheye/admin/login.do
--post-data="origUrl=&adminPassword=admin"
wget --load-cookies cookie.txt
--post-data="startRev=0&endRev=58&rep=2&action=rescan"
http://erdinger.sydney.atlassian.com/fisheye/admin/indexMaint.do
```

With that you could generate a post-revprop-change hook in svn that will update the repositories automatically.

How are indexing requests handled when they are triggered via commit hook

The indexing requests are performed in order of receipt by the usual incremental indexing threads. If there is already a pending request for indexing a particular repo, a subsequent request will be ignored.

Installation & Configuration FAQ

FishEye Installation & Configuration FAQ

- [Can I deploy FishEye or Crucible as a WAR?](#) — Unfortunately FishEye and Crucible can not be deployed as a WAR http://en.wikipedia.org/wiki/WAR_%28Sun_file_format%29.
- [Does Fisheye support SSL \(HTTPS\)?](#)
- [Improve FishEye scan performance](#) — Background information
- [Migrating FishEye Between Servers](#) — This page describes the process for migrating FishEye between servers.
- [Setting up a CVS mirror with rsync](#) — In situations where running FishEye on the same server as your CVS repository is not practical or possible, you can use the Linux utility 'rsync' to mirror the CVS repository contents onto the FishEye server.
- [What are the FishEye System Requirements?](#)
- [How to reset the Administration Page password in FishEye or Crucible](#)
- [How Do I Configure an Outbound Proxy Server for FishEye](#)
- [How to remove Crucible from FishEye 2.x or later](#)
- [How to run Fisheye or Crucible on startup on Mac OS X](#)

Can I deploy FishEye or Crucible as a WAR?

Unfortunately FishEye and Crucible can not be deployed as a [WAR](#). FishEye has some special needs for performance reasons that are not easily supported on third-party containers. While this is an often requested feature, there are no immediate plans to provide a WAR version of FishEye or FishEye+Crucible. However the upcoming separate edition of Crucible (i.e. without FishEye) may at some stage be available as a WAR.

Does Fisheye support SSL (HTTPS)?

FishEye has built-in SSL support from FishEye 2.4 onwards. Read [FishEye SSL configuration](#) for more information.

Improve FishEye scan performance

This page contains information about improving the performance of FishEye repository scans.

Background information

When you add a repository, FishEye needs to perform a once-off scan through the repository to build up its initial index and cache. This scan can take some time. Until this scan is complete, you may find that some data is not displayed. As a guide, FishEye should be able to process about 100KB-200KB per second on an averaged-size PC. If FishEye is accessing the repository over the network (e.g. over a NFS mount), then you should expect the initial scan to take longer.

General improvements

You can increase the speed of your scans using the following options:

- If your repository is non-local, set up a local repository mirror on the FishEye server. This will provide a major speed boost for anyone scanning a repository across a network.
- [Exclude](#) unused file types, unused directories and specific large files from FishEye.

Improve update performance during initial scan

One option is break large repositories into multiple smaller repositories. While this technique will not improve the overall initial scan time, it allows for all fully scanned repositories to be updated while the initial scan is still being performed on those remaining.

In FishEye 1.3.4 and later, the initial and incremental scans happen in separate, single threads. So, splitting the repositories will allow incremental scans to run concurrently alongside the initial scans. You may also wish to split projects into separate repositories, since permissions in FishEye are applied on a per-repository basis.

Improving initial scan performance for an SVN repository

svnsync is not an atomic protocol. It generally performs the commit as the svn sync user and then updates the revprops to match the original commit, under some circumstances, this can cause the

wrong user to be attributed as the author of a changeset. More information available at [Wrong User Reported When Indexing SVN Mirror Repository](#).

The http/s protocol has the slowest performance during the initial scan. The svn protocol (svn://) is faster and the file protocol (file://) is the fastest.

Therefore if you find your initial scan takes an extended amount of time (more than a day or two), you should consider switching over from the http/s protocol to the svn or file protocol to define the location of your SVN repository. (Use [svnsync](#) to mirror the repository onto the fisheye server, so that you can access it with the file protocol.)

```
E.g. Switch from
https://example.com/svn/project/
to
svn://example.com/svn/project/
or
file:///home/user/some/location/svn/project
```

In order for SVN protocol to work you need to have set up an [svnserve based server](#).

More information on how to troubleshoot SVN indexing related issues can be found [here](#).

Performance support

If you have implemented at least one of the suggestions above but are still experiencing slow performance, ask us for help:

1. First read the [Tuning FishEye performance](#) document.
2. Turn on debug logging using the [command line debug flag](#).
3. Allow FishEye to continue its initial scan overnight.
4. Create a new [support request](#) in the FishEye project, including your server environment and log files with the problem description.

Migrating FishEye Between Servers

This page describes the process for migrating FishEye between servers.

If you have defined the FISHEYE_INST [environment variable](#), then upgrades and migrations of your FishEye instance will be relatively simpler.

If you have defined FISHEYE_INST

1. Shut down your current FishEye server completely.
2. Copy the FISHEYE_INST directory to your destination server.
3. Copy and set up all of your [environment variables](#) from your source server to your destination server (remembering to set up your FISHEYE_INST directory to point to the location where you copied the data to in Step 2).
4. Install FishEye on your destination server.
5. Read the note about [external databases](#) below.
6. Start FishEye. It should pick up your environment variables, and from that access your FISHEYE_INST directory, which contains your configuration.

If you have not defined FISHEYE_INST but would like to set it up

- 1) Shut down your current FishEye server completely.
- 2) Copy the following three items into to a new folder on your destination server (for example, fisheye_inst):

- <FishEye installation directory>/config.xml

- <FishEye installation directory>/var
- <FishEye installation directory>/cache

3) Copy and set up all of your [environment variables](#) from your source server to your destination server. In addition to this, set up the FISHEYE_INST environment variable as follows, replacing the /path/to/fish eye_inst with the fully qualified path to the fisheye_inst folder you set up in Step 2:

```
export FISHEYE_INST=/path/to/fisheye_inst
```

4) Install FishEye on your destination server.

5) Read the note about [external databases](#) below.

6) Start FishEye. It should pick up your environment variables, and from that access your FISHEYE_INST directory, which contains your configuration. If you are using [JIRA for User Management](#) you might need to update the whitelist of the corresponding application connector in JIRA with the IP address of the new FishEye server to retain user authentication.

External Databases

If you are using an external database it must be accessible from the new server since the steps above only include installing a new application and relocating your configuration data and repository indexes.

If you wish to migrate both the application, data, *and* your external database to a different server (e.g. to create a development or test server) then you will need to make a copy of your external database and update the config.xml file (depending on the setup, located in either <FishEye/Crucible installation directory>/config.xml or FISHEYE_INST/config.xml) with the new database location **prior** to starting up the instance. See [Backing up and restoring FishEye data](#) for more information.

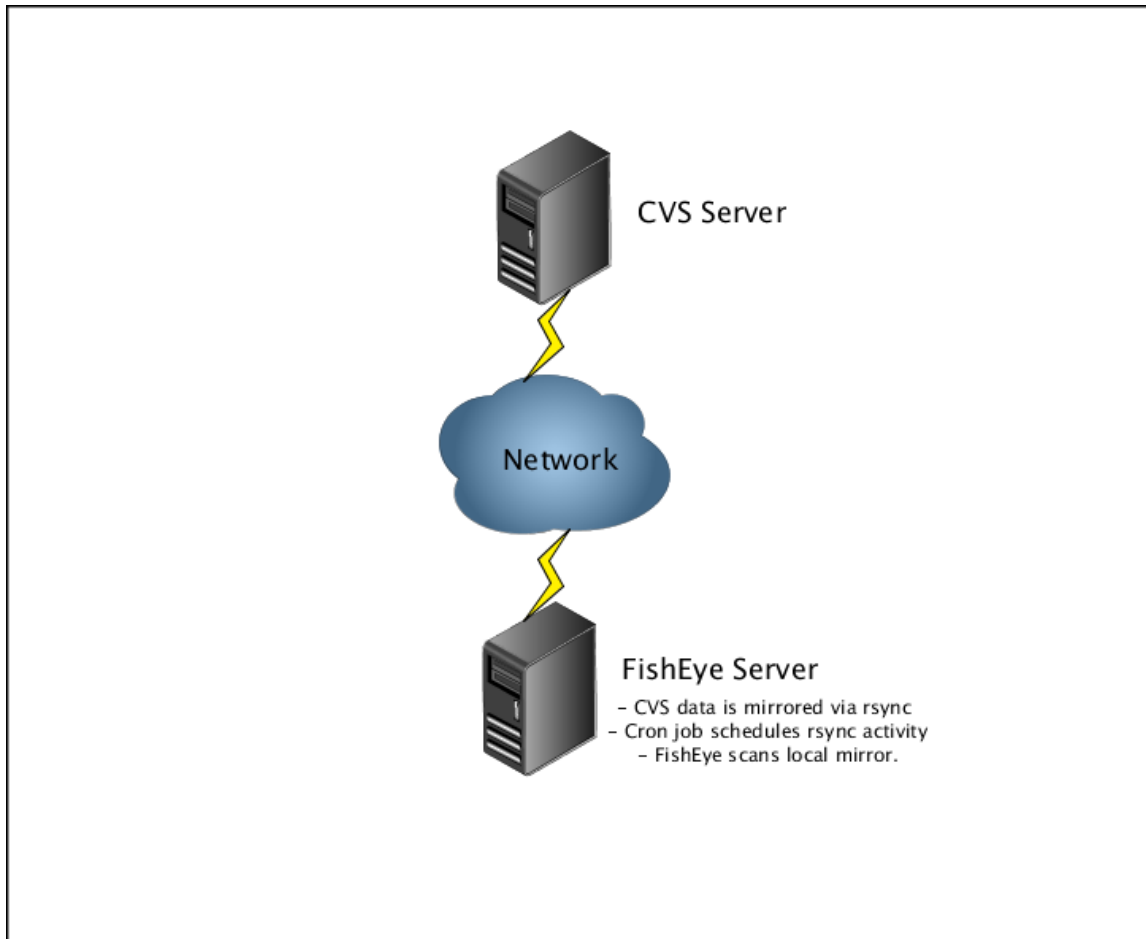
Setting up a CVS mirror with rsync

In situations where running FishEye on the same server as your CVS repository is not practical or possible, you can use the Linux utility 'rsync' to mirror the CVS repository contents onto the FishEye server. This is possible because CVS data is stored in a reasonably simple form in the file system.

We recommend this to achieve best performance when FishEye and CVS cannot be hosted on the same machine.

i This workaround requires the ability to SSH into both machines. Linux and Mac OS X operating systems have rsync built in. For Windows, you will need to [install rsync](#).

Diagram: A scenario where rsync is required



To set up a CVS mirror with rsync:

1. You will need to set up a local directory on the FishEye server for the mirrored CVS content, ensuring that this server has ample disk space to store the current CVS database and any future space requirements.
2. We will refer to the CVS instance (on your CVS Server,) as <CVS_HOME> and the new 'mirror directory' (on your FishEye server) as <MIRROR_HOME>.
3. Type the following rsync command on the console at the command-line of the FishEye server:

```
rsync --backup <CVS_HOME> <MIRROR_HOME>
```

A real-world example would look something like this:

```
rsync --backup user@cvserver:/CVS_server/path/to/instance  
/datastore/FishEye/cvs-mirror
```

4. Schedule the rsync command to run regularly with a [cron](#) job. Running hourly is a good default interval. Under Windows, use a native task scheduler.
5. With the cron job active, you will have established rsync to run an hourly comparison of the two directories and copy any changes across to the mirror directory as they occur. Note that running the rsync process will impact the FishEye server's performance (and also the CVS server's) to a certain degree.
6. In the FishEye admin interface, add the local 'mirror directory' as a new CVS repository and run the initial scan. As this is local data on the same file system, FishEye's scanning of this data will be optimal.
7. Adjust the [FishEye Updater](#) Full Scan period to one hour (the default is 15 minutes).
8. The rsync configuration is now complete. Monitor the disk space on both servers to ensure there is adequate headroom for the mirroring process.

For more information on the syntax for rsync, visit the [rsync home page](#).

What are the FishEye System Requirements?

Visit the FishEye [Supported platforms](#).

How to reset the Administration Page password in FishEye or Crucible

If you have forgotten or misplaced the password for the Admin page (`http://<FECRU_URL>:<FECRU_PORT>/admin/admin.do`), you can reset it manually.

To manually reset the admin password, please edit your `FISHEYE_INST/config.xml` file (make a backup as well before editing).

You will see something like:

```
<?xml version="1.0" encoding="UTF-8"?>
<config control-bind="127.0.0.1:8059" version="1.0"
admin-hash="352353256326369233A801FC3">
```

To reset the password to "**admin**", please change the admin-hash value so that it appears as `admin-hash="21232F297A57A5A743894A0E4A801FC3"`

```
<?xml version="1.0" encoding="UTF-8"?>
<config control-bind="127.0.0.1:8059" version="1.0"
admin-hash="21232F297A57A5A743894A0E4A801FC3">
```

Restart Fisheye for this to take effect. You should now be able to access the FishEye Admin page (`http://<FECRU_URL>:<FECRU_PORT>/admin/admin.do`) with the password "**admin**". Please change this password immediately from the Admin area: click **Change Admin password** (under "Security Settings").

See also [Best practices for FishEye configuration](#).

How Do I Configure an Outbound Proxy Server for FishEye

The Java Virtual Machine provides support for outbound proxy servers. To take advantage of this some additional parameters need to be passed to the JVM via the `FISHEYE_OPTS` environment variable:

```
export FISHEYE_OPTS="-Dhttp.proxyHost=proxy.example.org
-Dhttp.proxyPort=8080 -Dhttp.nonProxyHosts=*.foo.com|localhost
-Dhttp.proxyUser=username -Dhttp.proxyPassword=password"
```

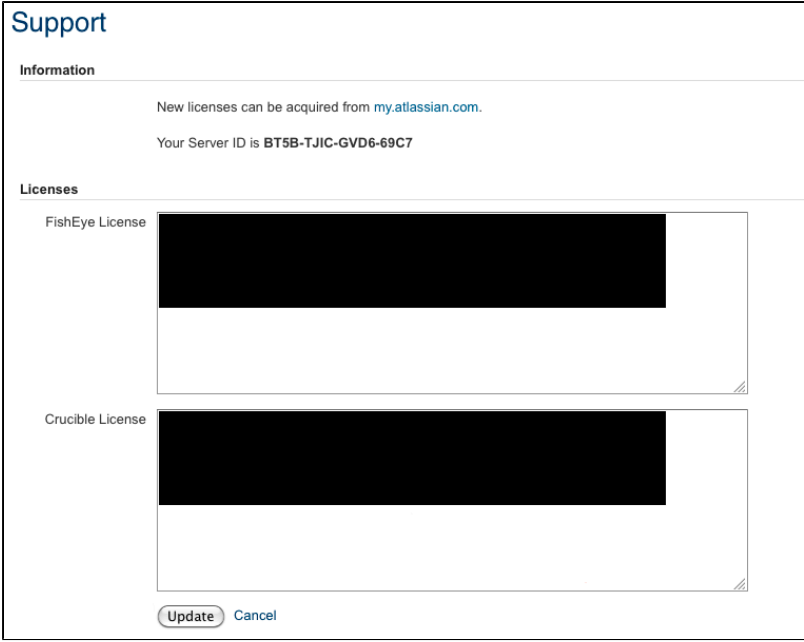
See [Environment variables](#) for instructions on how to set these parameters.

How to remove Crucible from FishEye 2.x or later

Use the same process to remove FishEye and keep Crucible; just remove the FishEye license instead.
Want to know [What happens if I decide to stop using FishEye with Crucible?](#)

Remove Crucible from a FishEye installation

1. Go to the Admin area (choose **Administration** from your user menu) and click **System Info** (under "System Settings").
2. Click **Edit License** (in the "License" section).
3. Remove the Crucible license from this screen:



Support

Information

New licenses can be acquired from my.atlassian.com.

Your Server ID is **BT5B-TJIC-GVD6-69C7**

Licenses

FishEye License

Crucible License

4. Click **Update** to save. Crucible will now be disabled and you will no longer see any reference to it in the application.

How to run Fisheye or Crucible on startup on Mac OS X

This article is only provided as a guide and has only been tested on Mac OS X 10.5

launchd does not provide support for service dependencies so if you are using an external database, this may not work for you. Fisheye assumes the database is available when it starts, and the startup scripts will not wait for the database to become available.

You need to create a .plist file to create items that will start at boot time. Please refer to <http://developer.apple.com/MacOSX/launchd.html> for details.

Here is an example .plist that should work for Fisheye/Crucible:


```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>Label</key>
  <string>com.atlassian.fisheye</string>
  <key>ProgramArguments</key>
  <array>
    <string>/path/to/<FishEye install directory>/bin/run.sh</string>
  </array>
  <key>OnDemand</key>
  <false/>
  <key>RunAtLoad</key>
  <true/>
  <key>StandardOutPath</key>
  <string>/path/to/FISHEYE_INST/var/log/fisheye.out</string>
  <key>StandardErrorPath</key>
  <string>/path/to/FISHEYE_INST/var/log/fisheye.out</string>
  <key>EnvironmentVariables</key>
  <dict>
    <key>FISHEYE_INST</key> <string>/path/to/FISHEYE_INST/</string>
    <key>FISHEYE_OPTS</key> <string>-Xms512m -Xmx1024m
-XX:MaxPermSize=128m</string>
  </dict>
</dict>
</plist>
```

Customize the `/path/to/FISHEYE_INST/` and `/path/to/<FishEye install directory>/` with the `FISHEYE_INST` and `FishEye` install directories respectively and make any required modifications to `FISHEYE_OPTS`. Save the file as `com.atlassian.fisheye.plist` and then try to load it with:

```
[amyers@erdinger:fecru-2.1.3]$ launchctl load
com.atlassian.fisheye.plist
[amyers@erdinger:fecru-2.1.3]$ launchctl start com.atlassian.fisheye
```

FishEye should now start up and you will be able to access it in your web browser.

Licensing FAQ

FishEye Licensing FAQ

- [Are anonymous users counted towards FishEye's license limits?](#) — Users accessing FishEye anonymously are, for all intents and purposes, unlimited users.
- [Restrictions on FishEye Starter Licenses](#)
- [Updating your FishEye license](#)
- [Git or Hg Repository exceeds number of allowed committers](#)

Are anonymous users counted towards FishEye's license limits?

The short answer is no. If you are using FishEye in your organization but most users require only anonymous access (that is, you have not set access restrictions on the content in your repositories), then an unlimited number of anonymous users can be accommodated regardless of the FishEye license you are using.

Users accessing FishEye anonymously are, for all intents and purposes, *unlimited* users.

However, if your users require permissions and controlled access to specific content in your repositories, then they will need to log in to FishEye. Hence, these users will need to create accounts and will be factored into the license limit.

Restrictions on FishEye Starter Licenses

This page explains the limitations of the FishEye Starter license and provides general information about using this license in production.

On this page:

- [What is a Starter License?](#)
- [What are the Starter License restrictions?](#)
- [What happens if I exceed the Starter License limits?](#)
 - [Evaluate](#)
 - [Upgrade](#)
 - [Reconfigure your repository](#)
- [Frequently Asked Questions](#)

What is a Starter License?

Starter licenses are low-cost licenses that allow small teams to make use of Atlassian products (see [more information](#)). FishEye Starter licenses were introduced with the release of FishEye 2.0.5 (October 2009).

What are the Starter License restrictions?

FishEye Starter Licenses are restricted to no more than:

- 5 FishEye users
- 5 repositories
- 10 historic [committers](#) per [repository](#)

See [FishEye Licensing and Purchasing](#) for more details.

Note that all other FishEye licenses support unlimited repositories and unlimited committers per repository.

What happens if I exceed the Starter License limits?

If you have more than 5 user accounts in FishEye, your users will not be able to log in to FishEye until an administrator reduces the number of user accounts.

If you have more than five repositories, FishEye will prevent more than five repositories from being enabled at any given time. Administrators can control which five repositories are enabled.

If you exceed more than ten [committers](#) in a [repository](#), the following warning will appear at the top of pages for the entire system:

NOTE: This repository, <repository-name> has more than 10 committers which exceeds the limits for your Starter license. Indexing has stopped. To fix this, you can [Evaluate](#), [Upgrade](#) or [Reconfigure your repository](#).

The links in this warning will lead you to the following solutions:

Evaluate

30-day evaluation licenses are available to allow you to [try out FishEye](#) and other Atlassian products. You can select a license that allows more users than your current license.

Upgrade

You can upgrade your license at any time (via the [Atlassian online store](#)), which will remove the [committer](#) and [repository](#) limits which apply to the Starter License. Please ensure to restart your repository, after the license upgrade, to ensure the changes are picked up for the new committer limit.

Reconfigure your repository

This option lets you configure your repository to remain within the limits of the Starter License. You can take the following actions to reduce the scope of FishEye's indexing:

1. **Change the repository definition to look at a subset of your repository**
Typically this involves setting the path within your repository that you wish FishEye to index. [Read more.](#)
2. **Exclude parts of the repository**
You can exclude portions of your repository that you are not interested in. [Committers](#) that are active in only these areas will not appear in FishEye and not be included in the committer count. [Read more.](#)
3. **Set a starting point**
Some of the FishEye SCM integrations allow you to configure a starting revision from which to start indexing. All [commits](#) prior to this starting point are not included in FishEye and do not contribute to the committer count. [Read more.](#)
4. **Map Committers**
If your developers have not correctly configured their committer names for Git or Mercurial, they may have committed with multiple identities. It is then possible to remap these to correct the problem. See the knowledge base article, [Git or Hg Repository exceeds number of allowed Committers](#)

Once you have reconfigured your repository, you will need to re-index the repository, allowing you to remain under the limits of the Starter license.

Frequently Asked Questions

Question	Answer
What happens when the 11th unique committer is encountered during indexing?	For all SCMs other than CVS FishEye will index all revisions up to but not including the revision that introduced the 11th committer. Since CVS is indexing is file-by-file based, FishEye will index files until it reaches the committer limit. Remaining files in the repository are not indexed. This means only files which have been indexed will be displayed in changesets and changesets may be incomplete.
What happens when a FishEye instance with a Starter license is started, using existing indexes with more than five repositories?	FishEye will only start indexing on the first five repositories. An administrator can use admin UI to adjust which repositories are enabled and hence control the five repositories that are started. FishEye should then be restarted.
What happens when a FishEye instance with a Starter license is started, using existing indexes with one or more repositories with more than ten committers?	FishEye will display all information currently indexed but for each repository that has reached the ten committer limit, no further revisions will be indexed.
What happens on upgrade from a Starter license, if indexing has been paused due to the committer limit being reached?	On restart of FishEye, indexing will resume for all repositories. Each repository can be restarted individually to avoid restarting FishEye. Due to the nature of CVS indexing, we recommend reindexing any CVS repositories which have reached the committer limit prior to the license upgrade.
What happens when upgrading from a Starter license, when repositories have not started due to the repository limit being reached?	On restart of FishEye, all enabled repositories will start. Each repository can be restarted individually to avoid restarting FishEye.

What happens if my evaluation license has expired and I upgrade to a Starter license, however I have exceeded the Starter license limitations?	As described above, a maximum of five repositories will start and for any repository with more than 10 committers, no further indexing will occur. All existing indexed content is retained and can be viewed.
What happens when downgrading to a Starter license, where the repository limit has been exceeded?	A maximum of five of your configured repositories will start running. The remainder will not start but will continue to be available.
What happens when downgrading to a Starter license, where the committer limit has been exceeded for one or more repositories?	No further indexing will occur for the repositories where the committer limit has been exceeded.

Updating your FishEye license

When you upgrade or renew your FishEye license, you will receive a new license key – you'll need to update your FishEye server with the new license.

Note that you can access your current license, or obtain a new license, by going to my.atlassian.com.

Related pages:

- [Upgrading from FishEye to Crucible](#)

To update your FishEye license key:

1. Log in to FishEye [Admin area](#).
2. Click **System Info** (under 'System Settings').
3. Click **Edit License** and paste your new license key into the appropriate text box.
4. Paste your new license into this box. Obtain a new license by clicking my.atlassian.com in the 'Information' section.
5. Click **Update**.

Support

Information

New licenses can be acquired from my.atlassian.com.

Your Server ID is **BMRI-3AS3-HMMD-PGXX**

Licenses

FishEye License

Evaluation

Crucible License

AAAA9g0ODAoPeNptkE9rwzAMxe/+FladPWa7hzXgQ5eELRAnpcmgH15Uo62G1A3+E9Zvv3Yt
dBs9C
IT03k8PPfS7RJvDRAWnfJ7NeMznNO96Kp64JAUG4+0Y7cGp3CdjtwNuMlpOMCQ4T0nu8acpl
Kl6e
xjnTAhirurH2hp0AfvjiA3siOnAGy6/R+uMvqPgDbdJ+i779eA/og
2KctP4TnA0X2CIEIIFdzOAiXbCy5kO/YS+KtSLXIVMLjrJ3rQu2PJ1vSZd2ahTsVqKZzGXklzz
n
+R1Vdzb3l+6TN7sIOD/v3wDPH51STAtAhRYkbBDq2j1NpXxLp5ws8aCihAgBwIVAJceVTPzGB
f4w
g5FWvNY7c5adD+EX02co

Update

Cancel

Git or Hg Repository exceeds number of allowed committers

Symptoms

Adding a Git repository to FishEye with a Starter license (10 users) installed will sometimes produce an error stating that there are more than 10 committers, even though there are less than 10 real people who have committed:

```
Repository index failed due to error - class
com.atlassian.fisheye.dvcs.handler.DvcsProcessRuntimeException:
com.cenqua.fisheye.LicensePolicyException: Exceeded number of allowed
committers
```

Cause

You have exceeded the number of committers allowed by the Starter License. This is either a legitimate occurrence (you have more than 10 actual committers) or this problem can also affect git and Mercurial Starter License users over time when their committers use different ids to commit to the repository.

Due to this change ([FE-2496](#) - User Mapping across multiple repositories needs to be easier CLOSED) in FishEye 2.6, when we scan a git repository, we now combine the author and email fields from the commit into a single field that FishEye uses to determine who the committer is. This allows automatic email mapping to map the commit by that committer to a FishEye User with the same email address.

Unfortunately, if your committers use different email addresses for their commits, they will now count as more than one committer, where previously they did not.

Resolution

For Mercurial, the only solution is to rewrite the history of your repository to map these committers to a single consistent committer id for each real committer you have (and to encourage them to be consistent in future).

This means getting all developers to delete their clones of the repository and re-clone once the history re-write is complete. Otherwise, they will probably end up pushing the old commits back into the repository being scanned by FishEye and the problem will reoccur.

Warning

Be very sure you know what you are doing during this process: you are re-writing history and the wrong command could lose all of your repositories history and your source code. Keep a backup clone of your repository in case something goes wrong.

Rewriting history is also valid for git (detailed below), but git has an easier alternative, using the `.mailmap` file. This allows the git repository to report to FishEye different names and email addresses than were actually used by the author - this will not affect any other repositories, unless they also get the `.mailmap` file in their local repositories. See the [git shortlog](#) man page for more details.

FishEye will use the version of the `.mailmap` file that is checked in at HEAD. This is normally the latest commit on master (or your default branch, if you have set it to other than master).

Once you set this up, you will need to go to the administration console in FishEye for your repository and under the 'Maintenance' tab, select 'Re-clone and Re-index'. Any changes you make to the `.mailmap` file will not take affect until you 'Re-clone and Re-index'.

Note that if you want to map multiple email addresses to one author, you need to specify them on multiple lines. E.g.

```
John Doe <john.doe@newcompany.com> <john@oldcompany.com>
John Doe <john.doe@newcompany.com> <johndoe@oldcompany.com>
John Doe <john.doe@newcompany.com> <jdoe@oldcompany.com>
```

Mercurial

In Mercurial, we use the [Convert Extension](#) to rewrite the committer names, using an `authormap` file. In the repository that FishEye is scanning, you will need to use the [Mercurial Queues Strip command](#) to remove all the current commits once the converted repository is ready, otherwise you will simply add a new set of commits next to the originals.

1. Get a copy of the repository and use:

```
hg log --template "{author}\n" | sort | uniq
```

to list the current committer names.

2. Identify those you want to rewrite and add an entry into a [authormap file](#) to rewrite them to another address.
3. Run:

```
hg convert my-clone rewritten-clone
```

4. [Strip](#) the repository that FishEye is indexing and push the `rewritten-clone` into it. Make sure all your developers use clones of the new copy of the repository from now on, otherwise they will possibly push back all the old commits and the problem will reoccur).
5. Go to the administration console in FishEye for your repository and under the '**Maintenance**' tab, select '**R e-clone and Re-index**'.

Git

This method will rewrite your commits with undesired author/emails in them, changing history. An easier approach (see above) is to use the .mailmap file as mentioned in the [git shortlog](#) man page. You will need to reclone and re-index after setting that up.

FishEye uses the git author name and email fields on a commit to identify who made the changes (which in FishEye is referred to as the "committer", but in git is the author; in git, the committer is who committed, possibly on behalf of the author), so we need to remap them.

1. To rewrite the committer names in git, use the [git filter-branch](#). You need to create a script to check for each committer you want to remap, and replace the name and email fields. The command needs to be run for each branch you have that you wish to modify:

```
git filter-branch --commit-filter '
    if [ "$GIT_AUTHOR_NAME" = "<Old Name>" ];
    then
        GIT_AUTHOR_NAME="<New Name>";
        GIT_AUTHOR_EMAIL="<New Email>";
    elif [ "$GIT_AUTHOR_NAME" = "<Other Old Name>" ];
    then
        GIT_AUTHOR_NAME="<Other New Name>";
        GIT_AUTHOR_EMAIL="<Other New Email>";
    fi;
git commit-tree "$@" Branch
```

Replace Branch with each branch you need to modify.

Note the warnings on the [git filter-branch](#) page.

2. Once the process is complete and your repository only contains the rewritten author names and email addresses, go to the administration console in FishEye for your repository and under the '**Maintenance**' tab, select '**Re-clone and Re-index**'.

Example EyeQL Queries

EyeQL
• How do I find changes made to a branch after a given tag? • How do I filter results? • How do I find changes between two versions, showing separate histories? • How do I find changes made between two version numbers? • How do I find commits without comments? • How do I find files on a branch, excluding deleted files? • How do I find files removed from a given branch? • How do I find revisions made by one author between versions? • How do I select the most recent revisions in a given branch? • How do I show all changesets which do not have reviews?

For more information on using EyeQL, see the [Reference guide](#).

How do I find changes made to a branch after a given tag?

Find changes made to Ant 1.5.x after 1.5 FINAL:

```
select revisions where on branch ANT_15_BRANCH and after tag
ANT_MAIN_15FINAL group by changeset
```

How do I filter results?

This query, finds files removed on the Ant 1.5 branch, but just returns the person and time the files were deleted:

```
select revisions where modified on branch ANT_15_BRANCH and is dead  
return path, author, date
```

How do I find changes between two versions, showing separate histories?

As above, but show the history of each file separately:

```
select revisions where between tags (ANT_MAIN_15FINAL, ANT_151_FINAL]  
group by file
```

How do I find changes made between two version numbers?

Find changes made between Ant 1.5 and 1.5.1:

```
select revisions where between tags (ANT_MAIN_15FINAL, ANT_151_FINAL]  
group by changeset
```

How do I find commits without comments?

Using the Advanced Search and EyeQL you can find commits that do not have comments using the following query:

```
select revisions from dir / where comment = "" group by changeset
```

How do I find files on a branch, excluding deleted files?

Find files on branch and exclude delete files:

```
select revisions where modified on branch ANT_15_BRANCH and not is  
deleted group by changeset
```

How do I find files removed from a given branch?

Find files removed on the Ant 1.5 branch:

```
select revisions where modified on branch ANT_15_BRANCH and is dead  
group by changeset
```

How do I find revisions made by one author between versions?

Find changes made by conor to Ant 1.5.x since 1.5.0:


```
select revisions where between tags (ANT_MAIN_15FINAL, ANT_154] and
author = conor group by changeset
```

How do I select the most recent revisions in a given branch?

Find Java files that are tagged ANT_151_FINAL and are head on the ANT_15_BRANCH: (i.e. files that haven't changed in 1.5.x since 1.5.1)

```
select revisions from dir /src/main where is head and tagged
ANT_151_FINAL and on branch ANT_15_BRANCH and path like *.java group by
changeset
```

How do I show all changesets which do not have reviews?

The following query will return any changesets that have not been reviewed.

```
select revisions where (not in any review)
```

Integration FAQ

The following FAQs relate to integrating FishEye with other Atlassian applications. You may also wish to refer to [Troubleshooting JIRA FishEye Plugin](#) in the FishEye Knowledge Base.

- [How do I disable the Source \(FishEye\) tab panel for non-code projects?](#)
- [How do I enable debug logging for the JIRA FishEye plugin?](#)
- [How do I uninstall the JIRA FishEye plugin?](#)
- [How is the Reviews \(Crucible\) tab panel for the JIRA FishEye Plugin populated?](#)
- [What do I do if I discover a bug with the JIRA FishEye plugin?](#)

How do I disable the Source (FishEye) tab panel for non-code projects?

By removing all users and groups from the 'View Development Tools' permission (called the 'View Issue Source Tab' permission prior to JIRA 6.1) in your project's permission scheme, the Source and Reviews tabs will be effectively disabled for that project.

How do I enable debug logging for the JIRA FishEye plugin?

For Plugin versions prior to 3.0

You can enable DEBUG logging for the JIRA FishEye plugin temporarily (until JIRA is shutdown) by accessing the following URL in a web browser when logged in as an administrator:

```
http://YOUR-JIRA-BASE-URL/secure/admin/ConfigureLogging.jspa?loggerName=
com.atlassian.jira.ext.fisheye&levelName=DEBUG
```

To enable debug logging across JIRA restarts, update the **CLASS-SPECIFIC LOGGING LEVELS** section of your WEB-INF/classes/log4j.properties file by inserting the following:

```
log4j.category.com.atlassian.jira.ext.fisheye = DEBUG, console, filelog
log4j.additivity.com.atlassian.jira.ext.fisheye = false
```

and restarting JIRA.

For Plugin versions 3.0+

You can enable DEBUG logging for the JIRA FishEye plugin temporarily (until JIRA is shutdown) by accessing the following URL in a web browser when logged in as an administrator:

```
http://YOUR-JIRA-BASE-URL/secure/admin/ConfigureLogging.jsps?loggerName=
com.atlassian.jirafisheyeplugin&levelName=DEBUG
```

To enable debug logging across JIRA restarts, update the **CLASS-SPECIFIC LOGGING LEVELS** section of your WEB-INF/classes/log4j.properties file by inserting the following:

```
log4j.category.com.atlassian.jirafisheyeplugin = DEBUG, console, filelog
log4j.additivity.com.atlassian.jirafisheyeplugin = false
```

and restarting JIRA.

How do I uninstall the JIRA FishEye plugin?

Plugin versions prior to 3.0

To uninstall, simply remove the jira-fisheye-plugin-*.jar from your WEB-INF/lib directory.

Plugin version 3.0+

To uninstall, simply remove any jira-fisheye-plugin-*.jar jars from your WEB-INF/classes/com/atlassian/jira/plugin/atlassian-bundled-plugins.zip and %JIRA_HOME%/plugins/installed-plugins directory.

How is the Reviews (Crucible) tab panel for the JIRA FishEye Plugin populated?

The Crucible 'Reviews' tab is populated differently depending on the version of the FishEye JIRA plugin you are using:

- **Prior to Version 2.3** of the plugin, a review is displayed on the Crucible tab panel if (and only if) it contains a revision that is displayed on the FishEye tab panel. Having the issue key in the description *will not* automatically link it to the issue.
- In **Version 2.3+**, you can configure the Crucible tab panel to use the method described above and/or *search for issue keys in the title/description of reviews*.

If you associate reviews with JIRA issues from the Crucible user interface, they will not necessarily show up in the issue tab panel. We are working on this issue (see [FISH-316](#)), but in the meantime, a workaround is to include the related issue key in the title of the review.

What do I do if I discover a bug with the JIRA FishEye plugin?

If you need assistance or think you've encountered a bug with the JIRA FishEye plugin, please raise an issue in the [JIRA FishEye Plugin project](#).

Subversion FAQ

FishEye Subversion FAQ

- [Configuring Start Revision based on date](#) — For Subversion repositories Fisheye has the ability to configure a Start Revision parameter to allow you to only index content from a given point in your repository.
- [Errors 'SEVERE assert' or 'Checksum mismatch'](#) — SVNKit may have problems with older version Subversion servers - versions 1.1.x and prior.
- [FishEye fails to connect to the Subversion repository after a short time of successful operation.](#) — On Unix systems, the svn:// protocol is usually handled by inetd or xinetd. These daemons apply, by default, a connection per second limit to incoming connections. Any connections above this rate are rejected by the server.
- [How can FishEye help with merging of branches in Subversion?](#) — In merge management, the main advantages of FishEye come from its search functionality. If you record the revisions merged when you check in a merge result, you can find this information in FishEye easily for the next merge operation.
- [Subversion Changeset Parents and Branches](#)
- [SVN Authentication Issues](#) — If multiple repositories have been defined in FishEye for the same SVN Server and those repositories use different credentials, FishEye may not use the correct credentials.
- [What are Subversion root and tag branches?](#)
- [Why do I need to describe the branch and tag structure for Subversion repositories?](#) — In Subversion, branches and tags are defined by convention, based on their path within a repository, and not directly defined by the repository. A few different layout alternatives are commonly used. It is also possible that you are using your own custom layout. As a result you need to describe to FishEye which paths in your repository are used as branches and tags.
- [Why don't all my tags show up in FishEye?](#)
- [About merges in Subversion](#)

Configuring Start Revision based on date

For Subversion repositories Fisheye has the ability to configure a **Start Revision** parameter to allow you to only index content from a given point in your repository. Quite often users will find it helpful to index from a revision on a given date. For example, you may want to only index SVN data in the past year. To determine the revision based on date, you can use the following command:

```
svn log -r {YYYY-MM-DD}:HEAD <SVN_URL> -l 1
```

The output of this command will reveal the revision number closest to the date that you provide.

Errors 'SEVERE assert' or 'Checksum mismatch'

When using SVNKit, you may see errors in the FishEye log such as 'SEVERE: assert #B' or 'Checksum mismatch'.

SVNKit may have problems with older version Subversion servers - versions 1.1.x and prior. If this is the case you should either use the native JavaHL layer or upgrade your Subversion server to a more recent version.

FishEye fails to connect to the Subversion repository after a short time of successful operation.

If you use the svn:// protocol to access a Subversion repository, you may notice that FishEye fails to connect to the repository after a short time of successful operation.

On Unix systems, the svn:// protocol is usually handled by inetd or xinetd. These daemons apply, by default, a connection per second limit to incoming connections. Any connections above this rate are rejected by the server.

Two options for fixing this problem:

- Ask your system administrator increase the connection rate allowed for the svn connection by updating the xinetd configuration, or
- Specify a connection per second limit in your FishEye repository definition, to prevent FishEye from exceeding the xinetd limits.

How can FishEye help with merging of branches in Subversion?

FishEye gives you a logical view of your branched files so you can see activity on a single file across multiple branches/trunk.

In merge management, the main advantages of FishEye come from its search functionality. If you record the revisions merged when you check in a merge result, you can find this information in FishEye easily for the next merge operation.

As an example, let's say you have a branch `dev` created at revision 1300 from `trunk`. Development has proceeded on both `trunk` and `dev`. At some point you wish to add the latest `trunk` changes into the `dev` branch. Let's say that is at revision 1400. When you check in the results of this merge, you would use some standard format checkin comment such as:

```
merge from trunk to dev 1300:1400
```

When you come to do the next merge, say at revision 1500, you can use FishEye search to find this checkin comment and know what the starting point for the merge should be. You can then check this in as:

```
merge from trunk to dev 1400:1500
```

Merges back to `trunk` from the `dev` branch are managed in the same way.

Subversion Changeset Parents and Branches

Why do some changesets have more than one branch? Why do these changesets have more than one parent?

In Subversion, a single changeset can have files and directories that are on different branches, as defined by the [SVN tag and branch structure](#). In this situation, the changeset is considered to be on all of the branches of its constituent file revisions. If a changeset is on more than one branch it can have a parent changeset of each of its branches, giving the changeset multiple parents.

FishEye does not track SVN merges, so merges are not indicated on the graph.

When I create a complex branch, how does FishEye determine which is its parent changeset? When I create a complex tag, how does FishEye decide which changeset to tag?

In Subversion, a *simple* branch or tag is created by copying a source directory, e.g. copying `/trunk` to `/branches/branch1` or `/tags/tag1`. The tag or branch is considered *complex* if a part of the copied directory is replaced with another version, e.g. `/trunk` is copied to `/tags/tag1`, and then `branches/branch1/dir1` is copied to `/tags/tag1/dir1`.

For the purpose of the [commit graph](#), FishEye looks at where the root directory was copied from, to determine where the branch or tag originated. In the example above, the label `tag1` would be applied to the latest changeset on `trunk` when the tag was created, even though part of the tag was copied from `branch1`. This only affects the annotation of the changeset, not the file revisions that are tagged — the tagged file revisions are still those on `trunk` or `branch1` as appropriate.

SVN Authentication Issues

If multiple repositories have been defined in FishEye for the same SVN Server and those repositories use different credentials, FishEye may not use the correct credentials.

FishEye does not control directly when authentication information is used to access Subversion repositories. It delegates this operation to the JavaHL layer in use. JavaHL will then ask FishEye to supply credentials when required, using a callback. The default JavaHL layer shipped with FishEye, SVNKit, can cache credentials at the server level rather than at the repository level.

There are different approaches to solving this issue:

Specify the username as part of the URL when defining your repository location (Recommended)

When using FishEye/Crucible 2.10, or later, we recommend that you simply include the username in the URL when defining the repository in FishEye. For example, use `http://username@host.com`.

Use the native JavaHL implementation

FishEye can be configured to [use the native JavaHL implementation](#), which will correctly apply the appropriate credentials.

Use the same authentication for the SVN server

The simplest solution is to have the same credentials for accessing the Subversion Server.

Use mock hostnames in the hosts file

Alternatively, SVNKit can be tricked into thinking that different servers are being used. For each connection to a repository a hostname in the *hosts* file can be defined.

All these entries then point to the same IP address of the SVN Server, but to SVNKit they look like different servers, thus bypassing the problem.

Example *hosts* entries (replace the IP address with the address of the SVN Server):

```
123.45.6.78 account1
123.45.6.78 account2
```

Replace these new server names in the SVN URLs:

```
http://account1/svn/project-a/
http://account2/svn/project-b/
```

What are Subversion root and tag branches?

FishEye identifies branches and tags in your Subversion repository by applying your specified [SVN tag and branch structure](#).

The "root:" branch

Any files or directories that fall outside the tag and branch structure are identified as being on the special branch, "root:". Some directories will almost always fall outside this structure. In general, root directories of branches are considered to be on the "root:" branch. This means that any changeset in which a branch is created is considered to be on branch, "root:". Additionally, any files or directories that fall outside the defined structure will be assigned branch, "root:". If you're seeing a lot of files and changesets on "root:", you may need to update your branch and tag structure in FishEye and rescan your repository, or exclude parts of your repository that don't follow your defined structure.

"tag:" branches

When FishEye detects that a tag has been created, it looks at the files that were tagged and adds the tag as an annotation to those file revisions. No file revisions are created at this point.

If a tag is modified after it has been created and committed, FishEye promotes the tag to a branch to preserve the history of the modification. For example, a user may create the tag "build1" by copying "trunk" to "tags/build1". If they then modify contents of tags/build1, a new branch "tag:build1" will be created for the modification.

Why do I need to describe the branch and tag structure for Subversion repositories?

In Subversion, branches and tags are defined by convention, based on their path within a repository, and not directly defined by the repository. A few different layout alternatives are commonly used. It is also possible that you are using your own custom layout. As a result you need to describe to FishEye which paths in your repository are used as [branches and tags](#).

It is very important that you correctly define in FishEye the layout you are using. If you do not, FishEye will not know which paths represent tags and branches. This will prevent FishEye from relating different versions of the same logical file across separate paths within your repository. It will also mean that FishEye's cache will be much larger as each tagged path will be indexed separately. This will result in an increase in the initial scan time and may reduce runtime performance.

If you are having trouble using Subversion tags, see [How tags work in Subversion](#).

Why don't all my tags show up in FishEye?

This page gives a detailed technical explanation of why certain issues affect Subversion users.

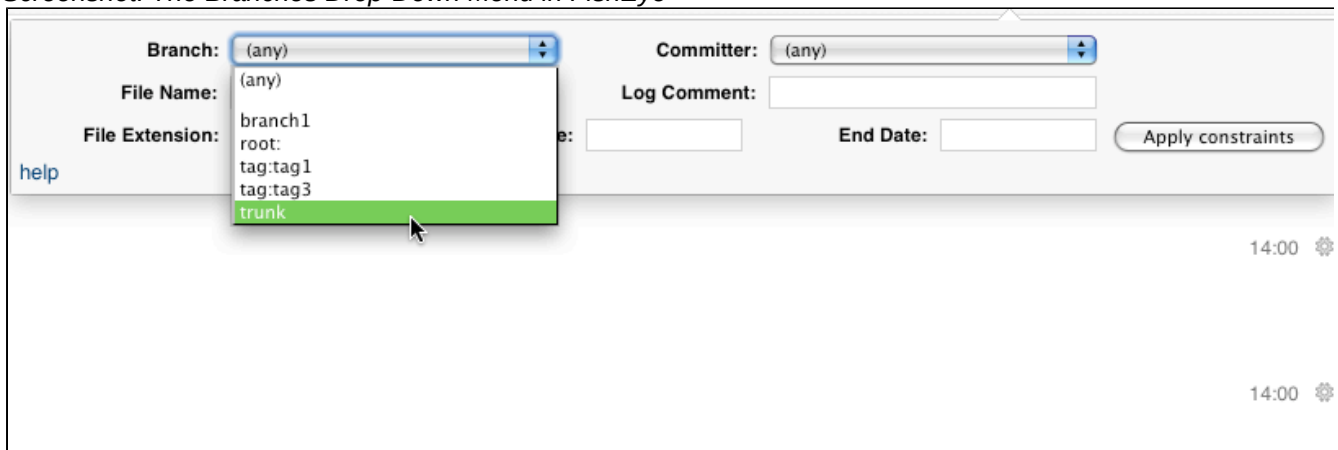
On this page:

- [Introduction](#)
- [How Subversion processes tags and branches](#)
- [An example from an Subversion repository](#)
- [Avoid modifications in the tag area](#)
- [Conclusion](#)

Introduction

When accessing Subversion from FishEye, you may see references to tags in the branches drop-down menu. In the example below, we can see **tag1** and **tag3** in the drop-down menu but not **tag2**:

Screenshot: The Branches Drop-Down Menu in FishEye



In actual fact, the branches drop-down menu shows only branch names. It does not show tags, but in some instances FishEye will synthesize a branch name to record certain operations. To understand how this occurs, you will need some background knowledge on Subversion tagging (introduced in the following segments of this page).

How Subversion processes tags and branches

In Subversion, tags are only a convention and are typically the result of a copy operation from the trunk to a tag area in the tags directory. When FishEye processes this copy operation, it recognizes that the destination is a tag directory and tags the source file on trunk with the name of the tag.

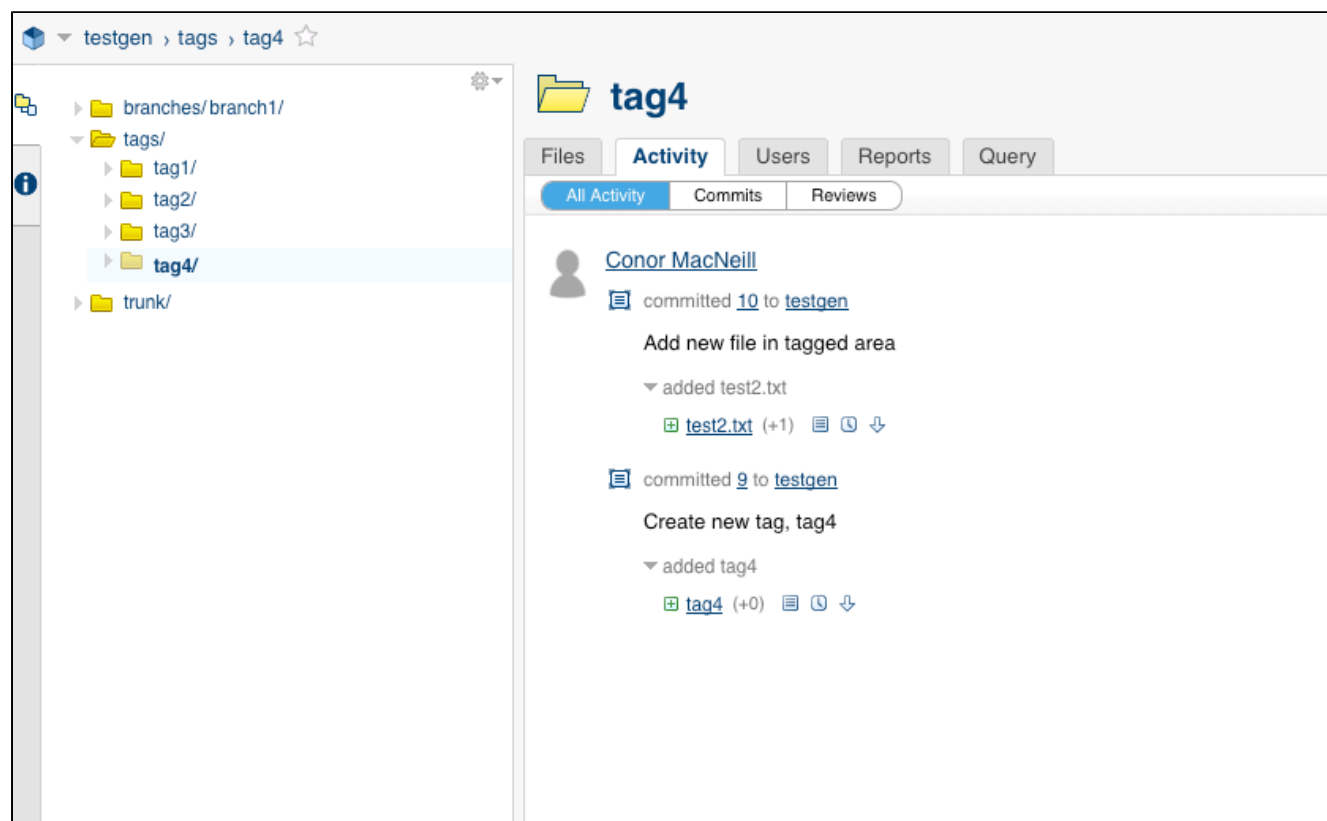
That is, FishEye is interpreting the Subversion copy to a tag directory as a tagging operation on the trunk files.

For regular changes in your Subversion repository, FishEye records each change against a branch where the change took place. If, however, after tagging, you make a change to a file in the tagged area, you are making a change outside trunk or a recognized branch. FishEye records such changes by creating an artificial branch name and associating that branch name with the change. The branch name is derived from the tag name by prepending "tag:" (in other words, the characters "tag:" appear as the first part of the name). The same thing will occur if you create a new file in the tagged area which does not come from an existing branch or trunk.

This is the reason you see some of your tags in the branch drop down. It means that for those tags, you have made a modification after the tagging operation.

An example from an Subversion repository

For example, consider **tag4** in this screenshot:



There are two changes here. The first creates the tag and the second adds a new file in the tagged area. This will result in the creation of an artificial branch, called "**tag:tag4**" within FishEye.

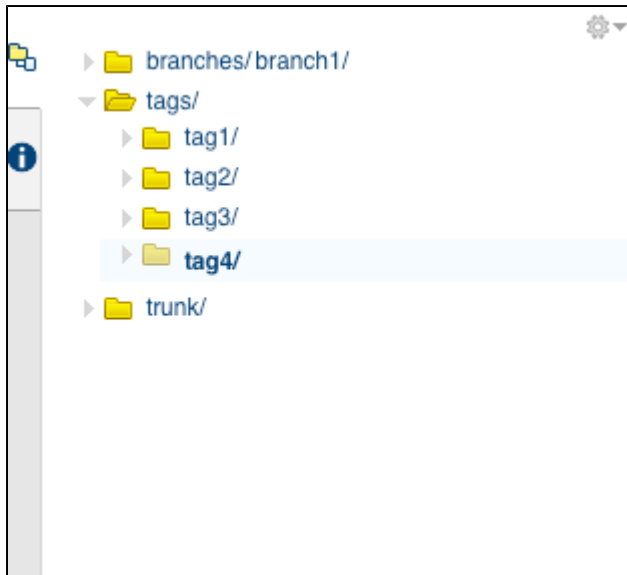
Avoid modifications in the tag area

In general, it's not good practice to make changes in the tag areas of a Subversion repository. Such changes can easily get lost if they are not applied to trunk or a current branch. It is preferable to make the change in trunk or a branch and then create a new tag to capture the update. Nevertheless, since Subversion tagging is merely a convention, this is sometimes convenient. FishEye handles this situation as described above.

Conclusion

In general a lot of systems have a large number of tags which would make the drop-down unworkable. This is the reason the tag field is a text-entry box below the branch drop-down menu in FishEye.

Since tags and branches are based on location convention in Subversion, the constraint is less effective than on other SCMs. You can always see the tag or branch you are interested in, based on its location in the repository. For example, the subdirectory list here shows all tags:



i If you want to constrain to a tag, enter the tag name in the tag field of the constraint filter.

About merges in Subversion

DO NOT PUBLISH BEFORE 4.1 release

This page explains how FishEye manages merge operations in Subversion, and provides examples of supported and unsupported merge formats.

- [Merge definition](#)
- [How FishEye shows merge information](#)
 - [Commit graph](#)
 - [Changeset view](#)
- [Acceptable path property format](#)
- [Merge targets and sources](#)
- [Unsupported operations](#)
 - [Advanced merging](#)

Merge definition

Subversion allows you to reapply changes made on other branches to the current working branch using the [merge](#) operation. For example, merging allows you to reintegrate a feature branch with trunk, or to reapply changes made on trunk to the current working branch. Subversion tracks this information in the dedicated `svn:mergeinfo` [path property](#). FishEye detects changes in those properties and uses those to mark parents of a changeset accordingly, effectively exposing merges between branches.

Notes:

- Although the path property might be set manually by using the [propset](#) command, FishEye trusts this information blindly, which means that if you add a required property in a correct format, FishEye will resolve parents for a changeset accordingly.
- The versions of the Subversion client and server used to perform merges both need to be version 1.5 or higher, since earlier versions do not have `svn:mergeinfo`. FishEye supports only branch and trunk path roots merges.

How FishEye shows merge information

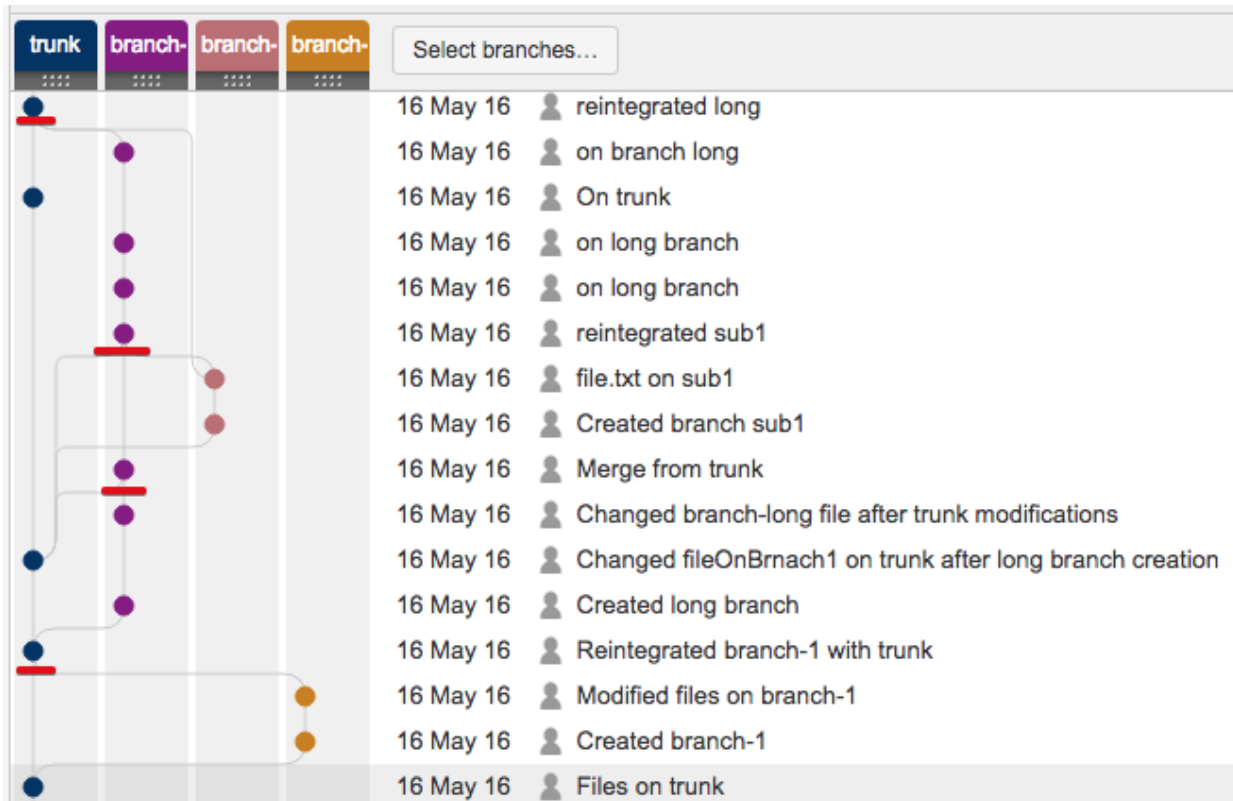
Changeset merges are exposed in three places:

- [commit graph](#)

- changeset view
- REST and Java API

Note: Although merge information is processed, changes on different branches are not reflected in the file ancestry history.

Commit graph



Changeset view

glewandowski committed 11 3 days ago
merge branch 2 to trunk

8 + 10
trunk

/trunk
file0.txt

View as Patch View in Activity Stream View in graph
Create Review

Blame j : next, k : prev

Acceptable path property format

FishEye uses the `svn:mergeinfo` property. It is set on a target path to which revisions has been merged and contains information about the revisions or revision range that has been merged, and the source of the merge. FishEye supports only base branches merge, which means that only the following format is acceptable:

(Source branch):(Range revision start)-(Range revision end)

For example:

```
/branches/feature-branch-1:10-15
```

However if none of the changesets in the range <10,15> made actual changes to feature-branch, (or those changes are not recognised as such by FishEye due to, for example, the included/excluded paths), the merge information won't be reflected.

Merge targets and sources

FishEye supports only branch and trunk path merges, which means that proper configuration of the [repository structure](#) is crucial. If the `mergeinfo` property is applied on a path that isn't identified as a branch or trunk root by SVN rules, it won't be processed. The same applies to `mergeinfo` value which describes the source of a merge. If it's not a branch/trunk root, or it's excluded by [include/exclude paths](#), it won't be processed or reflected in FishEye.

Considering the basic(built-in) rules and [recommended](#) repository structure, the following merge properties will be recognized:

```
Property changes on: branches/example-branch
```

```
Added: svn:mergeinfo  
Merged /trunk:r2-3
```

But the following **will not be recognized**:

```
Property changes on: trunk/directory
```

```
Added: svn:mergeinfo  
Merged /branches/example-branch/directory:r2-5
```

Unsupported operations

Advanced merging

None of the operations described [on this site](#) are supported by FishEye. Cherry picking revisions may result in `mergeinfo` formats similar to the following, which **won't be recognized**:

```
/branches/example-branch:5703  
/trunk:4622,5065,5300,5395,5403,5410-5414,5419,5423-5425
```

However, if you cherry pick revisions to create a valid revision range, it will be reflected in FishEye. For example, doing a couple of consecutive merge operations might result in a ranged `svn:mergeinfo` property format. Such a situation happens on the second merge command, which will transform `svn:mergeinfo` to contain the value `/example-branch:8-9`. In that case, FishEye will recognize it as a valid merge in the latest commit and mark it accordingly.

```
$ svn merge -c 8 ^/branches/example-branch && svn commit
$ svn merge -c 9 ^/branches/example-branch && svn commit
```

CVS FAQ

FishEye CVS FAQ

- [How does FishEye calculate CVS changesets?](#) — FishEye's goal is to allow changesets to be seen as a consistent stream of atomic commits. Revisions are collated into the same changeset provided that:
- [How is changeset ancestry implemented for CVS?](#)

How does FishEye calculate CVS changesets?

FishEye's goal is to allow changesets to be seen as a consistent stream of atomic commits. Revisions are collated into the same changeset provided that:

- They have the same commit comment.
- They are by the same author.
- They are on the same branch.
- The changeset does not span more than 10 minutes.
- The same file does not appear in a changeset more than once.

How is changeset ancestry implemented for CVS?

About Changeset Ancestry in FishEye

When FishEye indexes a CVS repository, it synthesizes a changeset identifier to group file-level changes into a single consistent changeset. The grouping is described in this FAQ: [How does FishEye calculate CVS changesets?](#).

Changeset ancestry was added in FishEye 2.6. Changeset ancestry refers to the linking of a changeset to a preceding/parent changeset(s). This allows you to view the development progress of your repository using the Commit Graph (see [Viewing the Commit Graph for a Repository](#)).

Changeset Ancestry for CVS

For CVS repositories, changeset ancestry is implemented, as follows:

- For all but the first change on a branch, FishEye chooses the most recent change on that branch as the parent changeset.
- For the first change on a branch, FishEye examines the branchpoints of all files in the branch and chooses the latest changeset that affected any such files as the parent changeset.

This approach ensures that a branch, whose first change is to a file which is very old, is not considered to have been branched at the time that file was last changed. It is considered to be branched at the last change to the repository instead.

Support Policies

Welcome to the support policies index page. Here, you'll find information about how Atlassian Support can help you and how to get in touch with our helpful support engineers. Please choose the relevant page below to find out more.

- [Bug Fixing Policy](#)
- [New Features Policy](#)
- [Security Bugfix Policy](#)

To request support from Atlassian, please raise a support issue in our online support system. To do this, visit support.atlassian.com, log in (creating an account if need be) and create an issue under FishEye. Our friendly support engineers will get right back to you with an answer.

Bug Fixing Policy

Summary

- Our Support team will help with workarounds and bug reporting
- We'll generally fix critical bugs in the next maintenance release
- We schedule non-critical bugs according to a variety of considerations

Report a bug

Building an add-on

Are you developing an add-on for an Atlassian product or using one of our APIs? Report any related bugs [here](#).

Bug reports

Atlassian Support is eager and happy to help verify bugs—we take pride in it! Create an issue in our [support system](#), providing as much information as you can about how to replicate the problem you're experiencing. We'll replicate the bug to verify, then lodge the report for you. We'll also try to construct workarounds if possible.

Search existing bug reports

Use our [issue tracker](#) to search for existing bugs, and [watch](#) the ones that are important to you. When you watch an issue, we'll send you an e-mail notification when the issue's updated.

How we approach bug fixing

Maintenance (bug fix) releases come out more frequently than major releases, and attempt to target the most critical bugs affecting our customers. The notation for a maintenance release is the final number in the version (the 1 in 6.0.1, for example).

If a bug is critical (production application down or major malfunction causing business revenue loss or high numbers of staff unable to perform their normal functions) we'll fix it in the next maintenance release, provided that:

- The fix is technically feasible (it doesn't require a major architectural change)
- It doesn't impact the quality or integrity of a product

For non-critical bugs, the developer assigned to fixing bugs prioritises the bug according to these factors:

- How many of our supported configurations are affected by the problem
- Whether there is an effective workaround or patch
- How difficult the issue is to fix
- Whether many bugs in one area can be fixed at one time

Developers responsible for fixing bugs also monitor comments on existing and new bugs, so you can comment to provide feedback if you need to. We give high priority to [security issues](#).

When considering the priority of a non-critical bug, we try to determine a *value* score for a bug. The score takes into account the severity of the bug from our customers' perspective, how prevalent the bug is, and whether new features on our roadmap may render the bug obsolete. Our developers combine the value score with a *complexity* score (how difficult the bug is) when selecting issues to work on.

Further reading

See [Atlassian Support Offerings](#) for more support-related information.

New Features Policy

Summary

- We encourage and display customer comments and votes openly in our issue tracking system, <http://jira.atlassian.com>.

- We do not publish roadmaps.
- Product Managers review our most popular voted issues on a regular basis.
- We schedule features based on a variety of factors.
- Our [Atlassian Bug Fixing Policy](#) is distinct from this process.
- Atlassian provides consistent updates on the top 20 issues.

How to track what features are being implemented

When a new feature or improvement is scheduled, the 'fix-for' version will be indicated in the JIRA issue. This happens for the upcoming release only. We maintain roadmaps for more distant releases internally, but because these roadmaps are often pre-empted by changing customer demands, we do not publish them.

How Atlassian chooses what to implement

In every [major release](#) we *aim* to implement highly requested features, but it is not the only determining factor. Other factors include:

- **Customer contact:** We get the chance to meet customers and hear their successes and challenges at Atlassian Summit, Atlassian Unite, developer conferences, and road shows.
- **Customer interviews:** All product managers at Atlassian do customer interviews. Our interviews are not simply to capture a list of features, but to understand our customers' goals and plans.
- **Community forums:** There are large volumes of posts on [answers](#), of votes and comments on [jira.atlassian.com](#), and of conversations on community forums like groups on LinkedIn.
- **Customer Support:** Our support team provides clear insights into the issues that are challenging for customers, and which are generating the most calls to support.
- **Atlassian Experts:** Our [Experts](#) provide insights into real-world customer deployments, especially for customers at scale.
- **Evaluator Feedback:** When someone new tries our products, we want to know what they liked and disliked and often reach out to them for more detail.
- **In product feedback:** The [JIRA Issue Collectors](#) that we embed our products for evaluators and our Early Access Program give us a constant pulse on how users are experiencing our product.
- **Usage data:** Are customers using the features we have developed?
- **Product strategy:** Our long-term strategic vision for the product.
- Please read our [post on Atlassian Answers](#) for a more detailed explanation.

How to contribute to feature development

Influencing Atlassian's release cycle

We encourage our customers to vote on issues that have been raised in our public JIRA instance, <http://jira.atlassian.com>. Please find out if your request already exists - if it does, vote for it. If you do not find it you may wish to create a new one.

Extending Atlassian products

Atlassian products have powerful and flexible extension APIs. If you would like to see a particular feature implemented, it may be possible to develop the feature as a plugin. Documentation regarding the [plugin APIs](#) is available. Advice on extending either product may be available on the user mailing-lists, or at [Atlassian Answers](#).

If you require significant customisations, you may wish to get in touch with our [partners](#). They specialise in extending Atlassian products and can do this work for you. If you are interested, please [contact us](#).

Further reading

See [Atlassian Support Offerings](#) for more support-related information.

Security Bugfix Policy

See [Security @ Atlassian](#) for more information on our security bugfix policy.

Troubleshooting

FishEye Troubleshooting

- **After I commit a change to my CVS repository, it takes a long time before it appears in FishEye.** — If you do not have a CVSROOT/history file, then a commit will not appear in FishEye until after FishEye has done a periodic full scan of your repository. You can configure the period of this scan in the Admin pages.
- **FishEye freezes unexpectedly** — If your FishEye 2.0 or 2.0.1 instance freezes unexpectedly, this could be caused by a known issue with FishEye and MySQL database technology.
- **I have installed FishEye, and the initial scan is taking a long time. Is this normal?** — As a guide, FishEye should be able to process about 100KB-200KB per second on an averaged-size PC. If FishEye is accessing the repository over the network (e.g. over a NFS mount), then you should expect the initial scan to take longer.
- **I have installed FishEye, but there is no data in the Changelog.** — When you add a repository, FishEye needs to scan through the repository to build up its index and cache. This scan can take some time. Until this scan is complete, you may find that some data is not displayed.
- **Initial scan and page loads are slow on Subversion** — It's possible that you've mis-configured your tag and branch structure and caused FishEye to process some or all files as trunk files. You should recheck your tag configuration.
- **JIRA Integration Issues**
- **Manually Generating a Thread Dump**
- **Message 'org.tigris.subversion.javahl.ClientException svn Java heap space'** — The Java heap space needs to be increased to an acceptable size. See the FishEye Tuning documentation for more information.
- **Problems with very long comments and MySQL migration** — There is a known issue with FishEye 2.0.x and very long comments when migrating your database to MySQL.
- **URLs with encoded slashes don't work, especially in Author constraints** — If the author names in your repository contain slashes or back-slashes, and you are using Apache, you may run into a problem where these characters are incorrectly escaped.
- **Manually Generating a Thread Dump (Draft)**
 - All Repositories fail - Could not execute query (MySQL database)

FishEye Knowledge Base

See the troubleshooting guides and technical announcements in the [FishEye knowledge base](#).

After I commit a change to my CVS repository, it takes a long time before it appears in FishEye.

If possible, FishEye will monitor and parse the CVSROOT/history file in your repository to quickly work out what has changed. You may want to check with your CVS administrator to ensure this feature of CVS is turned on.

If you do not have a CVSROOT/history file, then a commit will not appear in FishEye until after FishEye has done a periodic full scan of your repository. You can configure the period of this scan in the Admin pages.

FishEye freezes unexpectedly

Issue Symptoms

If your FishEye 2.0 or 2.0.1 instance freezes unexpectedly, this could be caused by a known issue with FishEye and MySQL database technology.

This issue manifests itself in some FishEye pages returning a server timeout error. To identify the issue, check the FishEye error log. For this issue, the following output will appear in the error log:

```
2009-07-15 15:34:45,555 ERROR [btpool0-519] fisheye.app
HibernateUtil-commitTransaction - Commit fail msg-0:Could not execute
JDBC batch update
2009-07-15 15:34:45,556 ERROR [btpool0-519] fisheye.app
HibernateUtil-commitTransaction - Commit fail msg-1:Lock wait timeout
exceeded; try restarting transaction
2009-07-15 15:34:45,557 ERROR [btpool0-519] fisheye.app
HibernateUtil-commitTransaction - Commit failed rolling back.
...
...
Caused by: java.sql.BatchUpdateException: Lock wait timeout exceeded;
try restarting transaction
    at
com.mysql.jdbc.ServerPreparedStatement.executeBatch(ServerPreparedStatement
ent.java:647)
    at
com.mchange.v2.c3p0.impl.NewProxyPreparedStatement.executeBatch(NewProxy
PreparedStatement.java:1723)
    at
org.hibernate.jdbc.BatchingBatcher.doExecuteBatch(BatchingBatcher.java:4
8)
    at
org.hibernate.jdbc.AbstractBatcher.executeBatch(AbstractBatcher.java:246
)
... 163 more
```

The FishEye error log can be found under FISHEYE_INST/var/log/fisheye-error.log.YYYY-MM-DD.

See the [JIRA issue](#) for more information.

Workaround

Until the issue is solved, the suggested course of action is to restart your FishEye instance. This will return FishEye to normal operation.

The FishEye development team is actively working on a solution and this be part of an upcoming point release of FishEye.

Requesting Support

If you require assistance in resolving the problem, please [raise a support request](#) under the FishEye project.

I have installed FishEye, and the initial scan is taking a long time. Is this normal?

When you add a repository, FishEye needs to scan through the repository to build up its index and cache. This scan can take some time. Until this scan is complete, you may find that some data is not displayed.

As a guide, FishEye should be able to process about 100KB-200KB per second on an averaged-size PC. If FishEye is accessing the repository over the network (e.g. over a NFS mount), then you should expect the initial scan to take longer.

For more details, see [Improve FishEye scan performance](#).

I have installed FishEye, but there is no data in the Changelog.

When you add a repository, FishEye needs to scan through the repository to build up its index and cache. This scan can take some time. Until this scan is complete, you may find that some data is not displayed.

As a guide, FishEye should be able to process about 100KB-200KB per second on an averaged-size PC. If

FishEye is accessing the repository over the network (e.g. over a NFS mount), then you should expect the initial scan to take longer.

Initial scan and page loads are slow on Subversion

Background Information

When you add a repository, FishEye needs to perform a once-off scan through the repository to build up its initial index and cache. This scan can take some time. Until this scan is complete, you may find that some data is not displayed. As a guide, FishEye should be able to process about 100KB-200KB per second on an averaged-size PC. If FishEye is accessing the repository over the network (e.g. over a NFS mount), then you should expect the initial scan to take longer. Read on if your scan appears to be considerably slower than expected.

Solutions

It's possible that you've mis-configured your tag and branch structure and caused FishEye to process some or all files as trunk files. You should [recheck your tag configuration](#).

If that fails, then the Atlassian support team will be happy to help you with this issue. Please [sign up for a support account](#) if you don't have one already, then login and [create a FishEye support request](#).

Users with very large or non-local repositories may be able to [improve their FishEye scan performance](#).

JIRA Integration Issues

Users are mapped to their own accounts when using [Trusted Applications](#).

If you (or the general account used for JIRA access, if not using Trusted Applications) do not have the permissions to carry out the JIRA actions linked from FishEye, an error will occur. Depending on the error returned from JIRA, FishEye may not display the error correctly or display it at all, simply reporting that "An error has occurred". To investigate what the error was, you can access the FishEye debug log, named `fisheye-debug.log.YYYY-MM-DD` under the `dist.inst/var/log` folder of your FishEye installation and look for the date and time when your error took place. Here, you will be able to follow the links and see what error the JIRA instance was producing by clicking through to JIRA.

Manually Generating a Thread Dump

If FishEye/Crucible stops responding or is showing poor performance, providing thread dumps to Support can help diagnose the problem.

Note: If you were asked by Atlassian technical support to create thread dumps please take at least six thread dumps – one every ten seconds for one minute so we can identify what the application is doing. Attach all generated files to the support ticket in addition to a Support Zip (the default Support Zip options includes `fisheye.out` which is the file to which thread dumps are sometimes written).

Generating a Thread Dump for Windows

Method 1: Windows scripts

We now have scripts for generating thread dumps externally on Windows. Download them from this [Bitbucket Repository](#).

Method 2: CTRL+BREAK (only if FishEye/Crucible is running in a console window)

1. Right click on the titlebar for the command console window where FishEye/Crucible is running to open the Properties dialog box.
2. Select the **Layout** tab.
3. Under Screen Buffer Size, set the **Height** to 3000 and click OK.
4. Press CTRL-BREAK on your keyboard.
5. This will output the thread dump to the command console.
6. Scroll back in the command console until you reach the line containing "Full thread dump".

7. Right click the title bar and select **Edit -> Mark**.
8. Highlight the entire text of the thread dump.
9. Right click the title bar and select **Edit -> Copy**.
10. Open a text editor and paste the thread dump and save the file.

Method 3 - VisualVM:

VisualVM is only provided as part of the Java Development Kit (JDK) distribution of Java. The JDK is not packaged with FishEye/Crucible, and will need to be downloaded separately. Your **JAVA_HOME** environment variable specifies which version of the Java Virtual Machine (JVM) FishEye/Crucible uses. If no **JAVA_HOME** environment variable is set, FishEye/Crucible uses the Java Runtime Environment (JRE) version packaged with the application.

Scenario 1: FishEye/Crucible is using the JVM provided by the JDK - AND - FishEye/Crucible is running as a console application:

1. Start **VisualVM** (<JDK installation directory>\bin\jvisualvm.exe)
2. Select **com.cenqua.fisheye.FishEyeCtl** application in the left-hand pane under **Local**.
3. Select the **Threads** tab in the right pane.
4. Click **Thread Dump**.
5. Select the **Threads** tab to return.

Scenario 2: FishEye/Crucible is not using the JVM provided by the JDK - AND - FishEye/Crucible is running as a console application:

1. Stop FishEye/Crucible.
2. Edit <FishEye/Crucible installation directory>\bin\fisheyectl.bat
3. Add the following JMX parameters to the **%_EXECCMD%** **%FISHEYE_OPTS%** property (within the existing quotes)

```
-Dcom.sun.management.jmxremote.port=6080
-Dcom.sun.management.jmxremote.authenticate=false
-Dcom.sun.management.jmxremote.ssl=false
```

You must choose a free port for JMX to bind to – this example uses port 6080

4. Restart FishEye/Crucible.
5. Start VisualVM: <JDK installation directory>\bin\jvisualvm.exe
6. Select **File > Add JMX Connection**.
7. Type: **localhost**:<your JMX port>
8. Select **com.cenqua.fisheye.FishEyeCtl** application in the left-hand pane under **Local**.
9. Select the **Threads** tab in the right pane.
10. Click **Thread Dump**.
11. Select the **Threads** tab to return.

Scenario 3: If FishEye/Crucible is running as a service:

1. Stop FishEye.
2. Go to **Start > Apps** (in Windows 8.1/Server 2012) or **All Apps** (in Windows 8) > **Configure FishEye**.
3. Add the following JMX parameters to the service configuration on the **Java** tab under **Java**:

```
-Dcom.sun.management.jmxremote.port=6080
-Dcom.sun.management.jmxremote.authenticate=false
-Dcom.sun.management.jmxremote.ssl=false
```

You must choose a free port for JMX to bind to – this example uses port 608

4. Click **Apply**, then **OK**
5. Restart FishEye.
6. Open Task Manager.
7. Look for the Process Name **Common Daemon Services Runner**.
8. Open the tree for the process and look for the sub-process "Atlassian FishEye". The PID for the process will be located under the PID column.

If you are running multiple Atlassian products on the same server, each will have a separate Process Name Common Daemon Services Runner.

9. Start VisualVM: <JDK installation directory>\bin\jvisualvm.exe
10. Select **File > Add JMX Connection**.
11. Type: **localhost**:<your JMX port> and click **OK**.
12. Select **localhost**:<your JMX port> in the left-hand pane under "Local".
13. Select the **Threads** tab in the right pane.
14. Click **Thread Dump**.
15. Select the **Threads** tab.

Generating a Thread Dump on Linux, including Solaris and other Unixes

Method 1 - jstack:

1. Find the process ID of FishEye/Crucible the JVM using the ps command.

```
FC_PID=`ps aux | grep -i fisheye | grep -i java | awk -F '[' '{print $2}'`;
```

2. Run the following command 6 times with a 10 second interval:

```
top -b -H -p $FC_PID -n 1 > fisheye_cpu_usage.`date +%s`.txt;
jstack -l $FC_PID > fisheye_threads.`date +%s`.txt
```

3. Look in the resulting CPU usage files (fisheye_cpu_usage.<timestamp>.txt) to identify which threads are consistently using a lot of CPU time.
4. Take the PID of the top 10 threads which are using CPU time and convert them to Hexadecimal (Eg: 11159 becomes 0x2b97).
5. Find the Hex values in the thread dump (fisheye_threads.<timestamp>.txt) files to figure out which threads are occupying CPU.

Method 2 - Kill Signal:

1. Find the process ID of FishEye/Crucible the JVM using the ps command:

```
ps aux | grep -i fisheye | grep -i java | awk -F '[' '{print $2}'
```

2. Run the following command 6 times with a 10 second interval:

The **kill -3** command will **NOT** terminate your FishEye/Crucible process.

```
kill -3 <pid>
```

The thread dump will be printed to FishEye/Crucible's standard output (fisheye.out).

Tools for analyzing thread dump files:

- Samurai - <http://samuraism.jp/samurai/en/index.html>
- Thread Dump Analyzer (TDA) - <https://java.net/projects/tda>

Message 'org.tigris.subversion.javahl.ClientException svn Java heap space'

When adding a new repository and on the initial scan, you may receive messages similar to this in the logs: `org.tigris.subversion.javahl.ClientException: svn: Java heap space`
The Java heap space needs to be increased to an acceptable size. See the [FishEye Tuning](#) documentation for more information.

Problems with very long comments and MySQL migration

Issue Symptoms

There is a known issue with FishEye 2.0.x and very long comments when migrating your database to MySQL. In some circumstances, this might result in truncation of very long comments, causing data loss.

Depending on your configuration, you may see an error message like this while migrating to MySQL, causing the migration to fail:

```
2009-07-16 16:56:12,390 ERROR [ThreadPool1] fisheye.app
com.cenqua.crucible.actions.admin.database.DBEditHelper-doGet -
Database migration failed:
java.sql.BatchUpdateException: Data truncation: Data too long for column
'cru_message' at row 1
java.sql.BatchUpdateException: Data truncation: Data too long for column
'cru_message' at row 1
```

 You may not see the message if you are running MySQL with default settings.

For more information, see the [JIRA issue](#).

Workaround

If your data contains very long comments or review descriptions (longer than 21,845 multibyte unicode characters), consider avoiding use of MySQL until this issue is resolved fully. Alternatively, use PostgreSQL or the default (built-in) HSQLDB database.

The FishEye developers are actively working on a solution to this problem and it will be addressed in an upcoming FishEye point release.

Requesting Support

If you require assistance in resolving the problem, please [raise a support request](#) under the FishEye project.

URLs with encoded slashes don't work, especially in Author constraints

If the author names in your repository contain slashes or back-slashes, and you are using Apache, you may run

into a problem where these characters are incorrectly escaped. By default Apache explicitly forbids encoded slashes or back-slashes in URLs. You can change this behavior with the following `httpd.conf` directive:

```
AllowEncodedSlashes On
```

This directive is documented [here](#).

FishEye Developer FAQ

This page contains answers to frequently asked questions posed by FishEye developers. For detailed information about developing in FishEye, see the [FishEye Developer documentation](#).

Feel free to comment, make submissions, or pose your own question on FishEye Development here.

- **Q:** *I'm getting the error "API access is disabled" as a response from <http://fisheye/api/rest/repositories> on my installation. How do I enable the API as a FishEye administrator?*
 ◀ [Click here to expand...](#)
A: A toggle to enable the API under "Server Settings" in the web admin interface existed in versions prior to 2.9 (see [Configuring the FishEye web server](#) for more details).
[See the FishEye 2.9 Release Notes](#)
- **Q:** *Is there any way to return unique results from an EyeQL query?*
 ◀ [Click here to expand...](#)
A: It is not currently possible to return unique results.
 An improvement request exists: [FE-1136](#). Your vote and comments on that issue are appreciated.

Contributing to the FishEye Documentation

Would you like to share your FishEye hints, tips and techniques with us and with other FishEye users? We welcome your contributions.

On this page:

- [Blogging your Technical Tips and Guides – Tips of the Trade](#)
- [Contributing Documentation in Other Languages](#)
- [Updating the Documentation Itself](#)
 - [Getting Permission to Update the Documentation](#)
 - [Our Style Guide](#)
 - [How we Manage Community Updates](#)

Blogging your Technical Tips and Guides – Tips of the Trade

Have you written a blog post describing a specific configuration of FishEye or a neat trick that you have discovered? Let us know, and we will link to your blog from our documentation.

Contributing Documentation in Other Languages

Have you written a guide to FishEye in a language other than English, or translated one of our guides? Let us know, and we will link to your guide from our documentation. [More...](#)

Updating the Documentation Itself

Have you found a mistake in the documentation, or do you have a small addition that would be so easy to add yourself rather than asking us to do it? You can update the documentation page directly

Getting Permission to Update the Documentation

Please submit the [Atlassian Contributor License Agreement](#).

Our Style Guide

Please read our short [guidelines for authors](#).

How we Manage Community Updates

Here is a quick guide to how we manage community contributions to our documentation and the copyright that applies to the documentation:

- **Monitoring by technical writers.** The Atlassian technical writers monitor the updates to the documentation spaces, using RSS feeds and watching the spaces. If someone makes an update that needs some attention from us, we will make the necessary changes.
- **Wiki permissions.** We use wiki permissions to determine who can edit the documentation spaces. We ask people to sign the [Atlassian Contributor License Agreement \(ACLA\)](#) and submit it to us. That allows us to verify that the applicant is a real person. Then we give them permission to update the documentation.
- **Copyright.** The Atlassian documentation is published under a Creative Commons CC BY license. Specifically, we use a [Creative Commons Attribution 2.5 Australia License](#). This means that anyone can copy, distribute and adapt our documentation provided they acknowledge the source of the documentation. The CC BY license is shown in the footer of every page, so that anyone who contributes to our documentation knows that their contribution falls under the same copyright.

RELATED TOPICS

[Author Guidelines](#)

[Atlassian Contributor License Agreement](#)

FishEye Documentation in Other Languages

Below are some links to FishEye documentation written in other languages. In some cases, the documentation may be a translation of the English documentation. In other cases, the documentation is an alternative guide written from scratch in another language. This page presents an opportunity for customers and community authors to share documentation that they have written in other languages.

Please be aware that these are external guides.

Most of the links point to external sites, and some of the information is relevant to a specific release of FishEye. Atlassian provides these links because the information is useful and relevant at the time it was written. Please check carefully whether the information is still relevant when you read it, and whether it is relevant to your version of FishEye. The information in the linked guides has not been tested or reviewed by Atlassian.

On this page:

- No guides yet

None
<i>No guides yet</i>
We do not yet have any guides to link here. Be the first to suggest one!

Adding Your Own Guide to this Page

Have you written a guide for FishEye in another language? Add a comment to this page, linking to your guide. We will include it if the content fits the requirements of this page.

Giving Feedback about One of the Guides

If you have feedback on one of the guides listed above, please give the feedback to the author of the linked guide.

If you want to let us know how useful (or otherwise) one of these guides is, please add a comment to this page.

Other Sources of Information

[FishEye documentation](#)
[Atlassian website](#)
[Atlassian blog](#)
[FishEye plugins](#)

FishEye Resources

Resources for Evaluators

- [Free Trial](#)
- [Feature Tour](#)

Resources for Administrators

- [FishEye Knowledge Base](#)
- [FishEye FAQ](#)
- [Guide to Installing an Atlassian Integrated Suite](#)
- [The big list of Atlassian gadgets](#)

Downloadable Documentation

- [FishEye documentation in PDF, HTML or XML formats](#)

Plugins

- [FishEye Developer Documentation](#)
- [Add-ons for FishEye](#)

Support

- [Atlassian Support](#)
- [Support Policies](#)

Forums

- [FishEye Forum](#)
- [FishEye Developers Forum](#)

Mailing Lists

- Visit <http://my.atlassian.com> to sign up for mailing lists relating to Atlassian products, such as technical alerts, product announcements and developer updates.

Feature Requests

- [Issue Tracker and Feature Requests for FishEye](#)

Glossary

Code repository or SCM ([Source Code Management](#)) software terminology can be confusing. This page provides definitions for some of the most commonly used terms.

On this page:

- [Branch](#)
- [Changeset](#)

- [Commit](#)
- [Committer](#)
- [CSID](#)
- [Head](#)
- [Linker](#)
- [Repository](#)
- [SCM](#)
- [Slurp](#)
- [Tag](#)
- [Trunk](#)

FishEye and its documentation uses the following terms:

Branch

A branch is an independent stream of work in a repository. For example, you might copy a set of files in the repository into a new branch, where you can carry on with a separate stream of work without cluttering up the main production area on trunk.

Different SCMs handle branching and merging in different ways. The common aspects allow users to create a branch in which to make changes which do not affect the files in other branches and the trunk development stream. These changes can then be merged into the trunk in a controlled fashion when a development unit of work is complete. Branches can also be used for experimental changes so that these do not affect the main development.

Changeset

A changeset is a collection of changes to files in a repository which are committed to the repository in a single operation with a single commit message. Not all SCMs support atomic commit operations. For these SCMs, FishEye will determine the file revisions which make up the changeset using a reliable heuristic (set of rules).

Different SCMs use different terms for the concept of a changeset, such as "changelist", which is generally interchangeable with changeset.

Commit

A commit is a single entry of content (usually source code) into a repository. It can be a single file or comprised of multiple file versions.

Committer

A committer is a user of an SCM repository who is adding content to the repository (where it will be permanently archived). Typically, a committer is a programmer who is committing source code but SCMs can also store other files such as documents, images and schematics.

CSID

An abbreviation for 'Changeset ID', this is a code that is used to reference every set of files that is committed to a repository. For example, if you commit three files to a repository, they are collectively a changeset, and will share the one CSID. CSIDs normally appear as a number, for example '**31905**'. In FishEye, CSIDs appear as links that you can click to visit the '**Changeset View**', which shows a list of the files in the left column, and the file contents or diffs in the right hand pane. In some circumstances you can hover your mouse over the CSID to see the '**Changeset Hover**' dialog, which displays the commit message, author, timestamp and files in the changeset.

Head

The "head" revision is the latest change to be made to a file in either a trunk or a branch part of a repository.

Linker

FishEye can render issue IDs or Bug IDs that appear in commit messages or comments as hyperlinks to the relevant issue/bug in your issue/bug tracker. The "linker" patterns used to detect the ID substrings can be configured separately for each repository. See [Linkers](#).

Repository

A repository is a area managed by an SCM where you store a set of related files, typically from a project or set of projects. The SCM is responsible for version controlling the files in the repository and maintaining their change history. A repository will contain the trunk and all branches for the files of the various projects. A single SCM instance can typically house multiple repositories.

SCM

SCM ([Source Code Management](#)) software is a category of computer software that archives complex sets of files, for example, all the source code comprised in a large multi-part software project. SCM systems keep copies of every version of every file, allowing you to completely restore and build any version of the software from any point in time.

Committers typically add new versions of code to the SCM once it is tested and approved by [peer code review](#) or quality assurance.

Each instance of an SCM can host multiple repositories.

Slurp

"Slurping" is a term that is synonymous with "repository scanning". FishEye must do intensive scans of the contents of repositories (SCM systems) in order to provide its lightning-fast web-based browsing of their contents. This can be referred to as a *slurp*, or *slurping*.

Tag

In SCM terminology, a "tag" is a label that is added to a number of files, to capture their collective state at a particular moment in time. A typical tag would be a specific software version number, that could be referenced to see all the files that belong to a specific version build of a software project.

Trunk

In SCM terminology, the "trunk" is the central part of the version control "tree". For example, you might copy a set of files in the repository into a new branch, where you can do new experimental work without cluttering up the main production area on trunk.

Collecting analytics for FishEye

We are continuously working to make FishEye better. Data about how you use FishEye helps us do that. We have updated our Privacy Policy so that we may collect usage data automatically, unless you disable collection. The data we collect includes information about the systems on which your installation of FishEye is operating, the features you use in FishEye, and your use of common IT terminology within the product. For more details, see our [Privacy Policy](#), in particular the 'Analytics Information from Downloadable Products' section.

See also our [End User Agreement](#).

How to change data collection settings?

You can opt in to, or out of, data collection at any time. A FishEye admin can change the data collection settings by going to **Analytics** (under 'Global Settings') in the FishEye admin area.

How is data collected?

We use the Atlassian Analytics plugin to collect event data in FishEye. Analytics logs are stored locally and then periodically uploaded to a secure location.